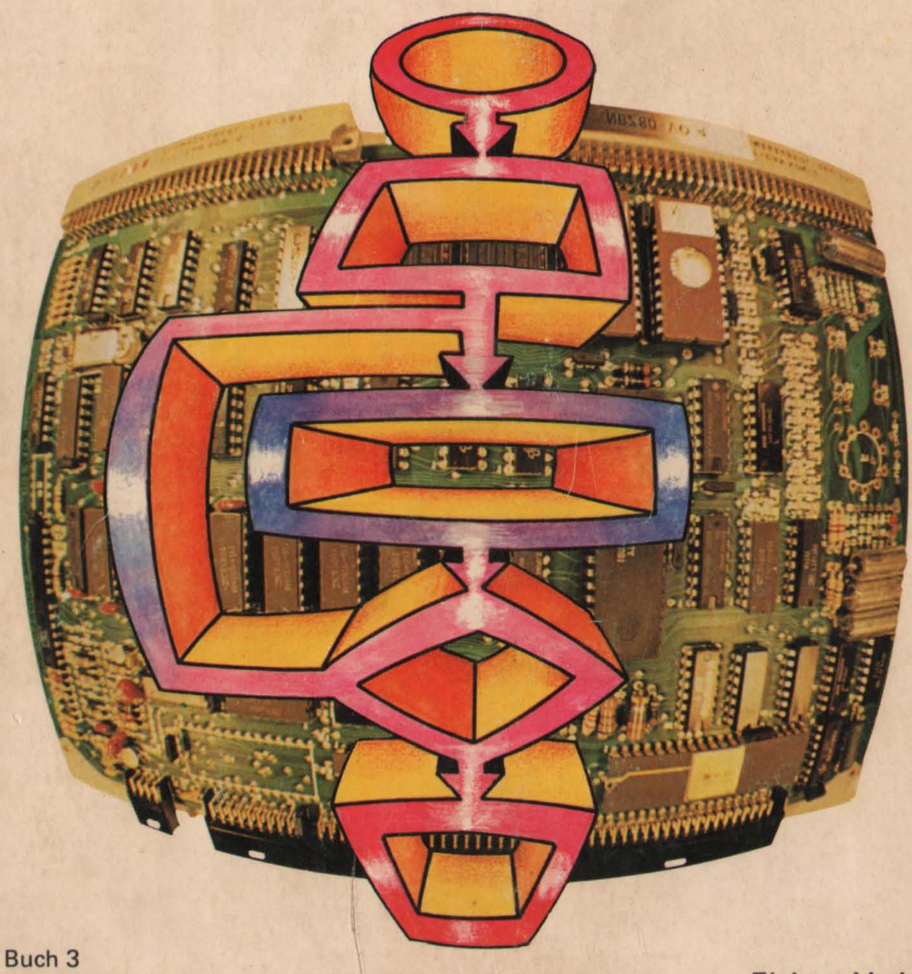
Z-80

Interface -Technik und Anwendung

Ein Trainingskonzept für moderne Mikroprozessor-Technik mit dem Nanocomputer NBZ80S.



Buch 3

Elizabeth A. Nichols, Joseph C. Nichols und Peter E. Rony

Elektor Verlag

Z-80

Interface-Technik und Anwendung

Buch 3

Ein Trainingskonzept für moderne Mikroprozessor-Technik mit
dem Nanocomputer NBZ80S

Elizabeth A. Nichols, Joseph C. Nichols und Peter E. Rony

Buchredaktion: R. Krings

ISBN 3-921608-17-1

Elektor Verlag GmbH
D-5133-Gangelt 1

Copyright ©1980 by Nanotran, Inc.

Die in diesem Buch veröffentlichten Beiträge sind urheberrechtlich geschützt. Ihre auch teilweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet.
Ein Haftung des Herausgebers für die Richtigkeit der veröffentlichten Aufsätze und Beiträge sowie für die Richtigkeit des technischen Inhalts und der Programme ist ausgeschlossen.

Vorwort

MIKROELEKTRONIK – Schlag- und auch manchmal Reizwort in der heutigen Gesellschaft. Wie dem auch sei: die neue Technik ist Realität und hat bereits ihren Stammplatz bei der Problembewältigung in vielen Lebensbereichen. Was noch im Zeitalter der voluminösen und energiefressenden Elektronenröhren Utopie war, ist heute Wirklichkeit. Die Entwicklung des Transistors hat diesen Aufschwung ermöglicht. Aufgrund der geringen Abmessungen und niedrigen Kosten war der Transistor das Basiselement der Computertechnik. Gatter, Flipflops, Zähler, Addierer und andere logische Schaltungen sind mit dem Transistor aufgebaut. Sicherlich ein Fortschritt, doch mittlerweile ist der Transistor von den ICs (integrated circuits; auch IS = integrierte Schaltung genannt) verdrängt. Die ICs vereinigen auf einen wizigen Chip eine Vielzahl von Bauelementen. Dabei nimmt die Packungsdichte noch stetig zu. Von 12 Transistoren im Jahr 1959 ist man heute bei mehr als 250.000 Bauelementen pro Chip angelangt. Und dabei sind die technischen Möglichkeiten noch nicht voll ausgeschöpft.
Es ist also wichtig, sich mit der Mikroprozessor-Technik anzufreunden. Dabei ist weniger die Frage nach der Hardware -dem schaltungstechnischen Konzept- als vielmehr die Frage nach der Software –dem Programm– interessant. Digitale Systeme werden in Zukunft nicht mehr durch Zusammenschaltung logischer Einzelbausteine, sondern durch entsprechende Programmierung eines Mikrocomputers realisiert. Das Fazit lautet also: Software anstelle fest verdrahteter Logik.
Die Z-80-CPU (CPU = central processing unit) ist ein 8-Bit Mikroprozessor. Zusammen mit einigen anderen ICs läßt sich bereits ein leistungsstarker Kleinstcomputer realisieren: der Nanocomputer NBZ80S.
Das vorliegende Buch ist das erste einer Trilogie; sie befaßt sich mit dem Mikroprozessor, der Programmierung sowie den Interface-Schaltungen.

Buch 3 beschreibt die Interface-Technik und deren Anwendung. Sie sammeln Hintergrundinformationen für den Umgang mit dem Z-80. Neben anderen Digital-Bausteinen ist der PIO (Parallele Ein-/Ausgabeeinheit) Mittelpunkt des Interesses. Nach der theoretischen Einführung je Interface-Detail folgt die Vertiefung des Erlernten im praktischen Experiment. Mit kleinen Programmen und Experimentier-Schaltungen wiederholt der Nanocomputer® NBZ80S (von SGS-ATES) die besprochenen Details. Schritt für Schritt zeigt der NBZ80S die praktische Wirkung. Diese Arbeitsmethode vermittelt solide Grundlagen der Programmier- und Interface-Technik.

Die Autoren bedanken sich bei R. Baldoni, A. Cattania, B. Facchi, F. Luraschi, C. Wallace, A. Watts, C. Edson, U. Broggi, J. Titus und D. Larsen für die freundliche Unterstützung bei der Erstellung dieses Buches.

Inhalt

ANHANG E

Kapitel 1.

Z-80 Interfacing

Interface – Grenzfläche (zweier Körper), Zwischenschicht, Trennfläche, Kopplungselektronik, Anschluß . . . – das sind einige Definitionen, die ein Wörterbuch, Deutsch – Englisch, unter dem Stichwort *interface* gibt. Doch damit allein läßt sich noch nicht viel anfangen. Für den Anwender von Mikrocomputern "steckt der Teufel im Detail". Deshalb geht das erste Kapitel ausführlich auf den Sinn und Zweck des eigentlichen "Interfacing" ein, nennt einige der wichtigsten Begriffe und befaßt sich schließlich auch mit den technischen Gegebenheiten.

Für Sie bedeutet dies konkret, daß Ihnen am Ende des Kapitels folgende Dinge klar sind:

- der Begriff *interface*. Sie können Beispiele für das "Interfacing" beim Mikroprozessor nennen;
- die Wichtigkeit von Interface-Schaltungen beim Mikrocomputer;
- einige technische Einzelheiten beim Interface des Z-80;
- die drei wichtigsten BUS-Leitungen (BUS = Sammelschiene, Hauptverbindung) bei einem Z-80 Standard-Mikrocomputer;
- die vier wichtigsten Interface-Kontrollsignale der Z-80-CPU (Zentraleinheit): $\overline{RD}$, $\overline{WR}$, $\overline{MREQ}$ und $\overline{IORQ}$;
- die Anschluß-Pinbelegung des Z-80-Mikroprozessors;
- die Bedeutung von Zeitdiagrammen für die Synchronisation der Z-80-CPU;
- die Bedeutung der Synchronisation. Dabei geht es um die Synchronisierung zwischen dem Interfacing und den I/O-Geräten (Ein/Ausgabegeräte) der CPU sowie den Speichern.

Was ist Interfacing?

Interfacing ist die Verbindung zum Teil unterschiedlicher Einzelteile zu einem funktionierenden und aufeinander abgestimmten Ganzen. (So fügen sich z.B. mehrere Menschen und Instrumente zu einer Band oder einem Orchester zusammen). Aufeinander abgestimmt heißt, daß z.B. alles synchron verläuft. Der Synchronismus spielt in der Mikrocomputer-Technik eine wesentliche Rolle. Deshalb seien nachfolgend einige Begriffe erläutert:

synchron – gleichlaufend oder gleichphasig. Das bedeutet z.B. im Eiskunst-Paarlauf den gleichen Bewegungsablauf bei beiden Läufern. In der

Computer-Technik bezieht es sich auf die Kontrolle einer Operationsfolge mit Hilfe des Taktsignals oder Taktimpulses.

Sychron-Computer — Bei dieser Computerart werden alle normalen Operationen und Befehle durch einen Taktgenerator kontrolliert.

Synchron-Logik — In diesem digitalen Logiksystem löst ein Taktimpuls den Befehl erst aus.

Synchron-Operation — Befehle, die in einem System durch ein Taktsignal kontrolliert werden.

sync — Kurzwort für synchron, Synchronisierung, Synchronisation usw.

synchronisieren — Zwei Elemente eines Systems aufeinander abstimmen bzw. in Gleichlauf bringen.

Synchronisationsimpuls — Der Synchronisationsimpuls kommt in der Regel von einem externen Gerät. Mit Hilfe einer Eingangsschaltung wird der Impuls in das System eingespeist und beide Geräte synchronisiert.

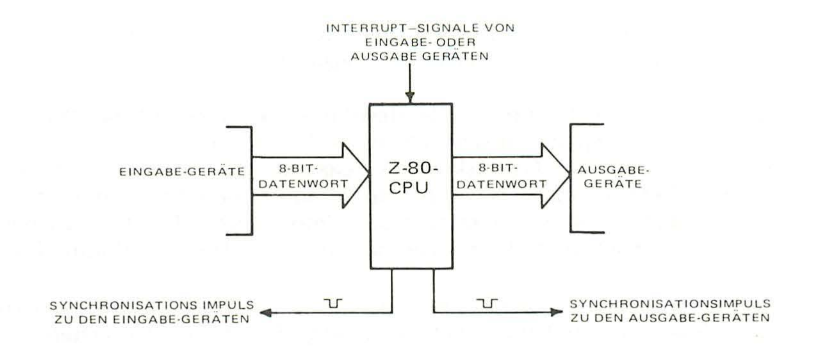
Synchron-Eingang — Der am Synchron-Eingang eines Flipflops (bistabile Kippstufe) vorhandene Impuls hat keinen direkten Einfluß auf den Ausgang. Erst wenn auch ein Taktsignal anliegt, ändert der Ausgang den logischen Zustand in eine bestimmte Richtung.

Aufgrund der gerade gegebenen Erklärungen kann man Computer-Interfacing definieren als "Synchronisation einer digitalen Datenübertragung zwischen Computer und externen Geräten einschließlich Speicher sowie der Ein- und Ausgabe-Geräte".

Obwohl sich bei verschiedenen Computer-Typen das Interfacing in einzelnen Details unterscheidet, ist das Prinzip überall gleich. Für den Z-80-Mikroprozessor zeigt Bild 1-1 die generellen Interface-Möglichkeiten. Es gilt:

- INPUT (Eingang): Überträgt Daten von einem externen Gerät zum Mikroprozessor.
- OUTPUT (Ausgang): Überträgt Daten vom Mikroprozessor zu einem externen Gerät.
- SYNCHRONISATIONSIMPULS: Koordiniert den Datentransfer zwischen Eingang und Ausgang des Mikroprozessors sowie den extern angeschlossenen Geräten.
- INTERRUPT (Unterbrechung): Definiert Interruptsignale und leitet sie von externen Geräten an den Mikroprozessor weiter.

Für die Datenübertragung zwischen der CPU und den externen Geräten steht ein Daten-BUS (Sammelschiene) zur Verfügung. Der Z-80-Daten-BUS kann acht Bitworte bidirektional (zweigleisig: in beiden Richtungen) übertragen. Das heißt, die Information kann über acht parallele Leitungen mit Hilfe des Carry-Bits zur Z-80-CPU gelangen oder von dort abgerufen werden. Das von der Datenübertagung betroffene externe Gerät erhält den Auswahlimpuls von der CPU über den Adreß-BUS. Die CPU adressiert die einzelnen externen Ein- und Ausgabe-Geräte mit den acht niederwertigsten Bits über den 16-Bit Adreß-BUS, während bei der Speicheradressierung der Adreß-BUS mit allen 16 Bits belegt ist. Die Carry-Signale synchronisieren die Daten- und Adreß-BUS-Informationen mit den Aktivitäten



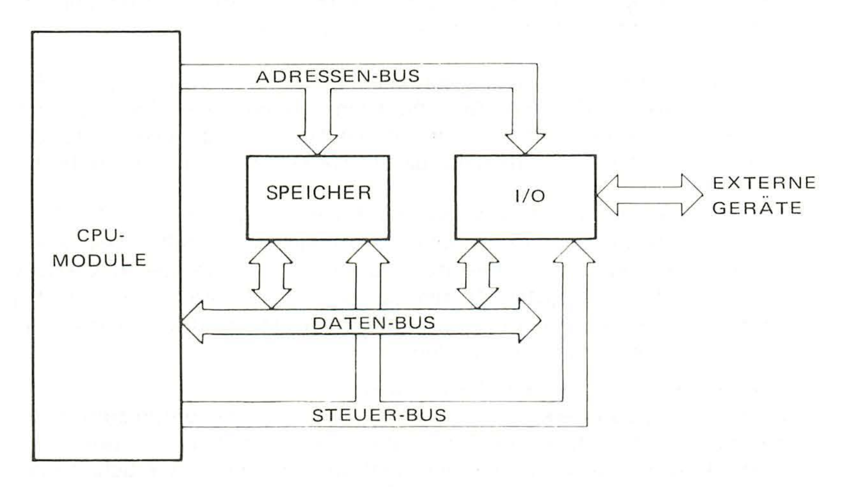


Bild 1-1. Das Blockschaltbild verdeutlicht die vier Interface-Aufgaben: Input, Output, Synchronisationsimpuls und Interrupt.

der CPU sowie der externen Geräte über den Steuer-BUS. Für den Austausch von Informationen zwischen der CPU und externen Geräten sind also drei BUS-Leitungen erforderlich: der Daten-BUS, der Adreß-BUS und der Steuer-BUS.

Das vorliegende Buch will Sie mit technischen Einzelheiten des Interfacing zwischen dem Z-80, dem Nanocomputer® und externen Geräten vertraut machen. Bevor jedoch auf Details eingegangen wird, stellt sich die Frage . . .

WARUM IST BEIM MIKROCOMPUTER DAS INTERFACING WICHTIG?

Eine Antwort auf diese Frage gibt der Artikel "The Smart Machine Revolution: Providing Products with Brainpower" (*publiziert in Business*

Week vom 5. Juli 1976, McGraw-Hill publication). Der Artikel ist im nachfolgenden nicht wörtlich, sondern nur sinngemäß wiedergegeben:

Nachdem die erste industrielle Revolution die menschliche Muskelkraft vervielfältigt hat, macht die zweite Revolution – der Mikrocomputer – dies mit der Geisteskraft. Das eigentliche der neuen Revolution ist der Mikroprozessor. Der Ein-Chip-Computer besteht aus nur einem wenige Quadratmillimeter großen Siliziumplättchen, das alle logischen und arithmetischen Funktionen enthält. Es ist das Herz des Computers.

Damit das Herz arbeiten kann, sind noch andere Funktionseinheiten (z.B. Speicher, Steuersysteme) notwendig. Das haben die ersten mit einem Mikroprozessor bestückten Geräte verdeutlicht. Erst alles zusammen macht den Mikroprozessor funktionsfähig.

Der Mikroprozessor soll im normalen Lebensbereich eines jeden Einzelnen seine häufigsten Anwendungen finden: das sind Haushalt, Arbeitsplatz und Kraftfahrzeug. So schätzt man, daß sich bis 1980 in jedem Haushalt 7 . . . 10 mikrogesteuerte Geräte im Einsatz befinden.

Durch die mannigfachen Anwendungsmöglichkeiten ist die Preistendenz bei den Herstellungskosten fallend. Infolgedessen stellen die Mikroprozessoren bei den intelligenten Maschinen und Geräten das Bauelement mit den niedrigsten Kosten dar, so daß auch die Geräte selbst zu einem äußerst niedrigen Preis erhältlich sind. Die niedrigen Kosten sind ein Garant für hohe Umsatzzahlen.

Die weitere Entwicklung der Mikroprozessoren:
Der Übergang von herkömmlichen Elektronik-Bauelementen zum Mikroprozessor (z.B. integrierte Schaltungen) hat die Konstruktions- und Herstellkosten gesenkt, weil er eine Vielzahl von integrierten Schaltungen und anderen Teilen ersetzt. Ist der Mikroprozessor einmal in ein System integriert, kann es enorme Marktvorteile bieten; die Funktionen eines Produktes brauchen nicht durch eine teure Neukonstruktion seiner Elektronik geändert zu werden, sondern einfach durch Abwandlung der Befehle oder Software, die im Speicher des Mikroprozessors aufbewahrt sind. Durch einen geringen Kostenaufwand entstehen neue Eigenschaften, und die neuen intelligenten Maschinen können Arbeiten verrichten, wie es zuvor in wirtschaftlicher Weise nicht möglich war.

Software ist nicht nur das größte Problem für den Anwender des Mikroprozessors, sondern auch das kostenintensivste. Ein Berater, der laufend intelligente Systeme für kleinere Firmen konzipiert hat, sagt dazu: "Die Software-Kosten sind bei einem Mikrocomputer noch höher als bei einem Minicomputer". Nach seinen Worten gibt er bis zu $ 100.000,– pro Jahr für den Aufbau der Software aus, während sich die Kosten für die Hardware lediglich auf ca. $ 20.000,– belaufen.

Die 1976 gemachten Voraussagen sind größtenteils eingetroffen: der Mikroprozessor hat sich in vielen Bereichen etabliert. Doch ohne Interface-Technik ist der Mikroprozessor nutzlos.

In einem früheren Abschnitt sind die vier wichtigsten Aufgaben des Interfacing, nämlich *input, output, Synchronisationsimpuls-Erzeugung* und *interrupt* bereits kurz beschrieben. Nachstehend die Definition einiger benutzter Fachausdrücke:

BUS: Ein Pfad, über den digitale Information aus jeder von mehreren Quellen zu jedem von mehreren Bestimmungsorten übertragen wird. Es läßt sich jeweils nur eine Information auf einmal übertragen. Während der Informationsübertragung müssen alle anderen am Bus angeschlossenen Quellen blockiert sein.

Bidirektionaler Daten-BUS: Ein Daten-BUS, über den Übertragung digitaler Information in beiden Richtungen möglich ist. Beim Z-80-Mikroprozessor-System ist es der bidirektionale Pfad, über den Daten zwischen CPU, Speicher und anderen externen Geräten übertragen werden.

Adreß-BUS: Ein BUS in einer Richtung, über den digitale Information übertragen wird, um entweder einen bestimmten Speicherplatz oder ein bestimmtes Eingabe-/Ausgabe-Gerät zu identifizieren. Der Adreß-BUS des Z-80-Mikroprozessor-Systems hat 16 Parallel-Leitungen.

Adresse: Eine Gruppe von Bits, die einen speziellen Speicherplatz oder ein externes Gerät identifizieren. Eine Z-80-Mikroprozessor-CPU benutzt zur einmaligen Identifikation eines Speicherplatzes 16 Bits, während eine 8-Bit-Adresse andere externe Geräte identifiziert.

Steuerwerk: Computereinheiten, die Befehle in der richtigen Reihenfolge ausführen, Befehle übersetzen und geeignete Synchronisations-Signale erzeugen.

Steuer-BUS: Eine Gruppe von Leitungen zur Übertragung von Signalen, die den Betrieb eines Mikrocomputer-Systems einschließlich Speicher und externer Geräte steuert. Diese Signale können von der CPU oder einem externen Gerät stammen. Der Steuer-BUS des Z-80-Mikrocomputer-Systems hat 13 Bits und überträgt unterschiedliche Signale. Es sind:
- die Synchronisationssignale für die Eingabe-/Ausgabe-Operationen zwischen der CPU, dem Speicher und externen Geräten;
- die CPU-Steuersignale für z.B. Interrupt, Warten, Halten;
- Steuersignale für den Zugang zum Adreß- und Daten-BUS.

I/O: Abkürzung für input/output (Eingabe/Ausgabe).

I/O-Gerät: Eingabe-/Ausgabe-Gerät. Dabei handelt es sich um z.B. Kartenleser, Magnetband, Drucker oder ein ähnliches Gerät, das Daten an einen Computer sendet bzw. von ihm oder einem Sekundärspeicher Daten empfängt. Im allgemeineren Sinne ist es jedes Digitalgerät einschließlich eines einzelnen Plättchens mit integrierten Schaltungen, das Daten oder Abtastimpule an einen Computer sendet oder von ihm empfängt.

CPU: Abkürzung für "central processing unit" = Zentraleinheit. Teil eines

Computer-Systems mit einem Steuerwerk, Rechenwerk und *speziellen* Register-Gruppen. Die CPU steuert die Befehlsverarbeitung, führt rechnerische Operationen durch und sendet Zeitsignale sowie andere Organisations-Operationen.

Zentraleinheit (Mikroprozessor): Ein einzelnes Plättchen mit integrierten Schaltungen, das Daten überträgt, Steuer-, Rechen-, Logik- und Interrupt-Operationen durchführt, indem es gespeicherte Befehle ausführt.

Speicher: Jedes Gerät, das logische 0- und 1-Bits so verarbeitet, daß man einzelne Bits oder eine Bit-Gruppe (Wort genannt) ein- und auslesen kann.

Die Z-80-Zentraleinheit (CPU)

Die erläuterten Definitionen finden Sie in den Blockschaltbildern 1-1 und 1-2 wieder. Besonders interessant ist das CPU-Modul. Es besteht aus einem 40-Pin-Chip und wird als Z-80-CPU bezeichnet. Die zur Z-80-CPU gehörende Befehlsgruppe ist bereits in Buch 1 beschrieben. Die folgenden Abschnitte befassen sich deshalb nicht mit der Z-80-Software, sondern mit der entsprechenden Hardware. Dazu gehört ein funktionelles Blockschaltbild, die Pinbelegung des Z-80 sowie einige Zeitdiagramme. Letztere verdeutlichen die Zusammenarbeit des Z-80 mit dem Speicher und anderen externen Geräten.

Bild 1-2 zeigt das funktionelle Blockschaltbild der Z-80-CPU.

Beim Ablauf eines gespeicherten Programms liest die CPU den Folgebefehl im Speicher ab. Dazu setzt der Programmzähler PC die gewünschte Adresse auf den Adreß-BUS. Gleichzeitig sind auf dem Steuer-BUS die entsprechenden Steuersignale vorhanden, um den Speicher zu aktivieren und dann die Daten auf dem Daten-BUS in das richtige Register innerhalb der CPU einzulesen. Dabei ist äußerst wichtig, daß die Zeiteinteilung stimmen muß. Das heißt: Der Speicherinhalt muß auf dem Daten-BUS sein, wenn die CPU den Daten-BUS abliest. Die CPU-Steuerfunktion koordiniert diese Aufgaben und gewährleistet, daß Befehls-Operations -Codes in das Befehlsregister eingesetzt und ordnungsgemäß entschlüsselt werden. Diese Funktion steuert auch die ALU(Arithmetic and Logic Unit), indem sie alle arithmetischen und logischen Operationen durchführt, die im Z-80-Befehlsvorrat enthalten sind. Zu diesen Operationen gehören Addieren, Subtrahieren, logisches UND, logisches ODER, exklusives ODER, Vergleichen, Links- oder Rechtsschieben und Rotieren, Werterhöhung, Wertverminderung, Bitsetzen, Bitrücksetzen, Bittesten. Bei der Durchführung dieser Operationen steht die ALU mit den 22 internen Registern, dem Befehlsregister und der Daten-BUS-Steuerung durch den internen Daten-BUS in Verbindung. Die Steuerungen für den Daten- und Adreß-BUS überwachen alle Aktivitäten in bezug auf den Datenaustausch zwischen CPU und Aussenwelt. Beachten Sie, daß der Daten-BUS bidirektional ist, während der Adreß-BUS einseitig gerichtet ist, und zwar von der CPU aus. Die CPU kann über den Adreß-BUS keine Daten empfangen. Bild 1-3 zeigt den Aufbau der CPU-Register.

8-BIT-
DATEN-BUS

8

DATEN-BUS-
STEUERUNG

BUS-STEUERUNG
CPU-STEUERUNG
SYSTEM-STEUERUNG

13

13-BIT-STEUER-BUS

BEFEHLSENT-
SCHLÜSSELUNG
UND
CPU-STEUERUNG

BEFEHLS-
REGISTER

INTERNER DATEN-BUS

ALU
(ARITHMETIC AND
LOGIC UNIT =
ARITHMETISCHE UND
LOGISCHE EINHEIT)

INTERNE
STEUERUNG
DER CPU

CPU-
REGISTER

ADRESS-BUS-
STEUERUNG

16

16-BIT-
ADRESS-BUS

+5v
GND
Ø

Bild 1-2. Funktionelles Blockschaltbild.

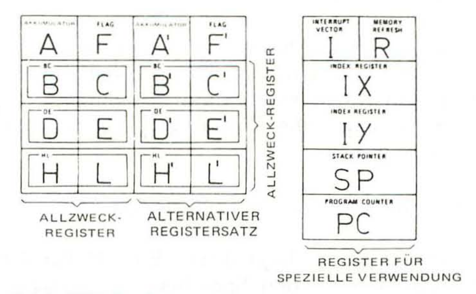


Bild 1-3. Aufbau der Z-80-CPU-Register

Die kleinen Kästchen stellen 8-Bit-Register, die großen Kästchen 16-Bit-Register dar.

Bild 1-4 zeigt die Pin-Belegung der Z-80-CPU. Dabei sind besonders der Adreß-BUS, der Daten-BUS sowie das System-Control (Steuersignale des Systems) von Bedeutung. Die folgenden Abschnitte gehen näher auf diese Signale ein ohne sie jedoch bis in die letzte Einzelheit zu erklären. Das wäre an dieser Stelle auch verfrüht.

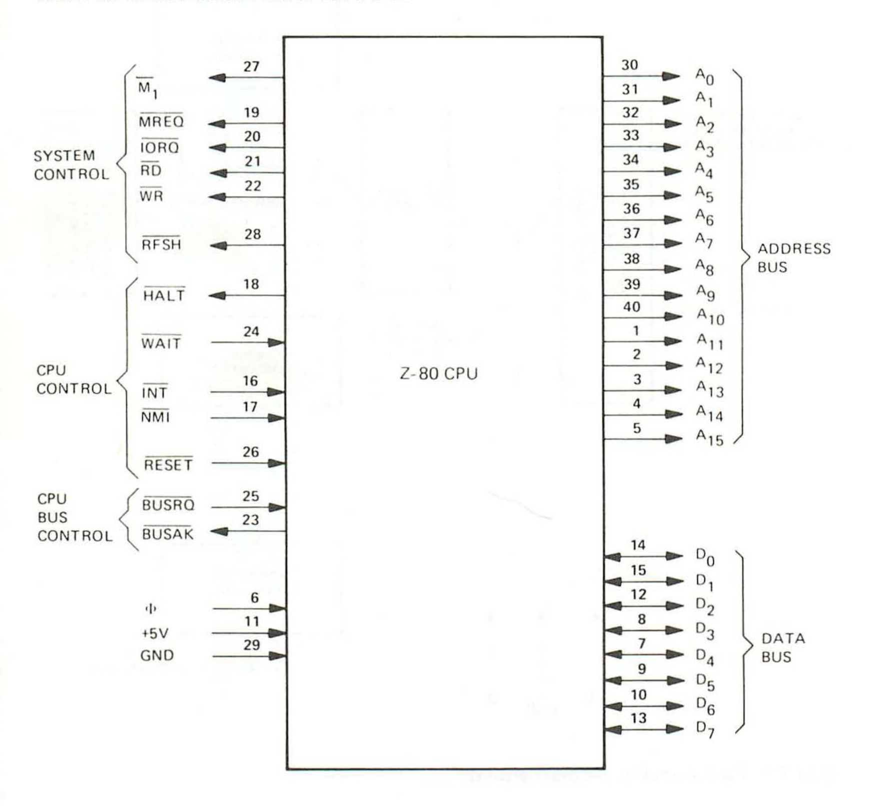


Bild 1-4. Pin-Belegung der Z-80-CPU

Im Verlaufe des Buches werden die einzelnen Signale an den entsprechenden Stellen ausführlich beschrieben. Sie sind lediglich bei der Pin-Belegung bereits erwähnt, um Ihnen einen Eindruck davon zu geben, was Sie erwartet. Aber keine Angst, die Z-80-CPU ist eben auch nur ein IC, allerdings mit 40 Anschlußpins.

Die einzelnen Pins haben folgende Funktionen:

A0-A15 (Adreß-BUS) Tri-Ausgang, high-aktiv. Ein 16-Bit-Adreß-BUS, der die Adressen für den Speicher (bis zu 64k Bytes = 2^{16}), den Datenaustausch sowie für den Datenaustausch des Ein-/Ausgabegeräts liefert. Für die Ein-/Ausgabe-Adressierung

werden die acht niederwertigen Adreßbits genommen, damit der Benutzer bis zu 256 = 2^8 Ein- oder Ausgänge direkt wählen kann. A0 ist das niederwertigste Adreßbit. Während der Refreshzeit (refresh = erneuern, auffrischen) enthalten die niederwertigen 7 Bits eine gültige Refresh-Adresse (Speicher-Refresh, wird in Kapitel 3 erörtert).

D0-D7
(Daten-BUS)
Tri-State-Ein-/Ausgang, high-aktiv, ein bidirektionaler 8-Bit-Daten-BUS, der zum Austausch von Daten zwischen Speicher und Ein-/Ausgabegeräten benutzt wird.

$\overline{\mathrm{M1}}$
(Maschinenzyklus Eins)
Ausgang, low-aktiv. Zeigt an, daß der augenblickliche Maschinenzyklus der Operationscode-Abrufzyklus einer Befehlsausführung ist. Bei der Ausführung von 2-Bit-Operationscodes wird $\overline{\mathrm{M1}}$ erzeugt, während jeder Operationscode abgerufen wird. Diese 2-Bit-Codes beginnen immer mit CB, DD, ED oder FD (hex). $\overline{\mathrm{M1}}$ tritt auch mit $\overline{\mathrm{IORQ}}$ auf, um einen Interrupt-Quittungszyklus anzuzeigen.

$\overline{\mathrm{MREQ}}$
(Speicher-Anforderung)
Tri-State-Ausgang, low-aktiv, zeigt an, daß der Adreß-BUS eine gültige Adresse für eine Speicher-Operation (Lesen oder Schreiben) enthält.

$\overline{\mathrm{IORQ}}$
(Ein-/Ausgangs-Anforderung)
Tri-State-Ausgang, low-aktiv, zeigt an, daß die niederwertige Hälfte des Adreß-BUS eine gültige Ein-/Ausgabe-Adresse für eine Ein-/Ausgabe-Operation (Lesen oder Schreiben) enthält. Das $\overline{\mathrm{M1}}$-Signal erzeugt ein $\overline{\mathrm{IORQ}}$-Signal, wenn ein Interrupt quittiert wird um anzuzeigen, daß der Daten-BUS einen Interrupt-Antwort-Vektor aufnehmen kann. Interrupt-Quittungs-Operationen entstehen während der M1-Zeit; in dieser Zeit laufen keine Ein-/Ausgabe-Operationen ab.

$\overline{\mathrm{RD}}$
(Lesen)
Tri-State-Ausgang, low-aktiv, zeigt an, daß die CPU Daten vom Speicher oder einem Ein-/Ausgabegerät ablesen soll. Das adressierte Ein-/Ausgabegerät bzw. der Speicher interpretiert dieses Signal als Aufforderung, Daten auf den CPU-Daten-BUS zu legen.

$\overline{\mathrm{WR}}$
(Schreiben)
Tri-State-Ausgang, low-aktiv, zeigt an, daß der CPU-Daten-BUS gültige Daten enthält, die in dem angesprochenen Speicher oder Ein-/Ausgabegerät gespeichert werden sollen.

$\overline{\mathrm{RFSH}}$
(Refresh)
Ausgang, low-aktiv, zeigt an, daß die niederwertigen 7 Bits des Adreß-BUS eine Refresh-Adresse für dynamische Speicher enthalten und das laufende $\overline{\mathrm{MREQ}}$-Signal zur Einleitung eines Refresh-Zyklus für alle dynamischen Speicher zu benutzen ist.

$\overline{\text{HALT}}$ (Halt-Zustand)
Ausgang, low-aktiv, zeigt an, daß die CPU einen (Software-) HALT-Befehl ausgeführt hat und zur weiteren Abarbeitung des Programms auf ein Interrupt-Signal wartet (ein nichtmaskierbares oder freigegebenes maskierbares Interrupt). Im HALT-Zustand führt die CPU zur Sicherstellung des Refresh-Vorgangs Leerbefehle aus.

$\overline{\text{WAIT}}$ (Wartesignal)
Eingang, low-aktiv, zeigt der Z-80-CPU an, daß der angesprochene Speicher oder die Ein-/Ausgabegeräte noch nicht zur Datenübertragung bereit sind.

$\overline{\text{INT}}$ (Interrupt-Anforderung)
Eingang, low-aktiv, das entsprechende Signal erzeugen die Ein-/Ausgabegeräte. Eine Anforderung wird nach Abarbeitung eines in Ausführung befindlichen Befehls berücksichtigt, soweit das interne softwaregesteuerte Interrupt-Freigabe-Flipflop gesetzt und das $\overline{\text{BUSRQ}}$-Signal nicht aktiv ist. Wenn die CPU das Interrupt annimmt, folgt zu Beginn des nächsten Befehlszyklus ein Quittungssignal ($\overline{\text{IORQ}}$ während der M1-Zeit). Die CPU kann ein Interrupt auf drei verschiedene Arten beantworten, die zu einem späteren Zeitpunkt erläutert sind.

$\overline{\text{NMI}}$ (Nichtmaskierbares Interrupt)
Ausgang, getriggerte Negativflanke, hat höhere Priorität als $\overline{\text{INT}}$ und wird am Ende der laufenden Anweisung stets anerkannt, unabhängig vom Zustand des internen Interrupt-Freigabe-Flipflops. Zwingt die Z-80-CPU automatisch, die Programmbehandlung bei der Speicheradresse 0066H wiederaufzunehmen. Der Programmzähler wird automatisch im äußeren Stapel zwischengespeichert, so daß der Benutzer zu dem unterbrochenen Programm zurückkehren kann. Beachten Sie, daß kontinuierliche Wartesignal-Zyklen die laufende Anweisung daran hindern kann aufzuhören und daß $\overline{\text{BUSRQ}}$ Vorrang vor $\overline{\text{NMI}}$ hat.

$\overline{\text{RESET}}$ (Rückstellen)
Eingang, low-aktiv, setzt den Programmzähler auf 0 und die CPU auf den Anfangszustand.
1) Sperrung des Interrupt-Freigabe-Flipflops;
2) Setzen des Registers I auf 00H;
3) Setzen des Registers R auf 00H;
4) Setzen der Interrupt-Betriebsart auf 0.
Während des Rückstellvorgangs befinden sich der Adreß- und Daten-BUS im hochohmigen und alle Ausgangs-Steuersignale im inaktiven Zustand.

$\overline{\text{BUSRQ}}$ (BUS-Anforderung)
Eingang, low-aktiv, bringt CPU-Adreß-BUS und Tri-State-Ausgangs-Steuersignale in einen hochohmigen Zustand, damit andere Geräte die BUS-Leitungen steuern können. Wird $\overline{\text{BUSRQ}}$ aktiviert, versetzt die CPU sie in einen hochohmigen Zustand, sobald der laufende CPU-Maschinenzyklus beendet ist.

$\overline{\text{BUSAK}}$ (BUS-Bestätigung) — Ausgang, low-aktiv, bestätigt dem anfordernden Gerät, daß sich die CPU-Adreß-BUS-, Daten-BUS- und Tri-State-Steuer-BUS-Signale in den hochohmigen Zustand befinden und das externe Gerät diese Signale jetzt steuern kann.

Φ — Einphasige TTL-Takteingabe

Z-80-Befehlszyklen: Maschinen- und T-Zyklen

Was geschieht nun innerhalb des Z-80-Chips und seiner 40 Pins während der Ausführung eines Befehls? Jeder Z-80-Befehl besteht aus einer Reihe von Grundoperationen oder *Maschinenzyklen.* Die Z-80-CPU kann nur sieben Grundoperationen oder Maschinenzyklen ausführen:

1. Operationscode-Abrufanweisung (M1-Zyklus)
2. Speicherzyklus (Datenlesen oder -schreiben)
3. Ein-/Ausgabezyklus (Lesen oder Schreiben)
4. BUS-Anforderungs/Bestätigungszyklus
5. Interrupt-Anforderungs/Bestätigungszyklus
6. Nichtmaskierbarer Interrupt-Anforderungs/Bestätigungszyklus
7. Ausgang von einer HALT-Anweisung

Lediglich die ersten sechs Zyklen beziehen sich klar und direkt auf die vier Hauptaufgaben des Interfacing.
Der Rest dieses Kapitels sowie einige der nächtsten Kapitel befassen sich ausschließlich mit den drei ersten Arten von Maschinenzyklen.
Bei der Ausführung eines Befehls wie z.B. des LD A, 00H (hex: 3E 00), der im Speicherplatz 0100 gespeichert ist, führt der Z-80 einen M1-Zyklus aus, um den Objectcode (Operationscode) des Befehls vom Speicher abzurufen. Dies hat zur Folge, daß die Adresse 0100 auf den 16-Bit-Adreß-BUS gelangt und die $\overline{\text{MREQ}}$- und $\overline{\text{RD}}$-Steuersignale (siehe Pin 19 und Pin 21) aktiviert werden. Das Speichergerät unterbricht diese Signale wie folgt:

$\overline{\text{MREQ}}$ – low aktiv (negative Logik) bedeutet, das ein Zugriff zum Speicher hergestellt ist.

$\overline{\text{RD}}$ – low-aktiv (negative Logik) bedeutet, daß es sich bei diesem Zugriff um eine Lese-Operation handelt. Die Adresse 0100 auf dem Adreß-BUS bestimmt die abzulesende Speicherstelle.

Der Speicher gibt den Inhalt der Stelle 0100 auf den Daten-BUS. Die CPU liest den Daten-BUS ab und speichert seinen Inhalt. Betrifft die Lese-Operation das erste Byte eines Befehls, wird das $\overline{\text{M1}}$-Signal aktiviert. Es zeigt an, daß es sich bei dem Byte um einen Objektcode handeln muß. Für Befehle mit zwei Objektcodes werden zwei $\overline{\text{M1}}$-Zyklen ausgeführt, um sie vom Speicher zu erhalten. In genannten Beispiel findet ein $\overline{\text{M1}}$-Zyklus statt, um den Objektcode 3E einzulesen.

Der Z-80 entschlüsselt den Code 3E als Ladeadresse unmittelbar zum Akkumulator-Befehl. Er führt anschließend einen Speicher-Maschinenzy-

klus (Lesen) aus, um den Inhalt des Speicherplatzes 0101 in den Akkumulator einzugeben. Eine Speicherablesung stimmt fast genau mit einem $\overline{M1}$-Zyklus überein, nur daß das $\overline{M1}$-Signal inaktiv und die Zeitberechnung etwas anders ist.
Bevor Sie Zeitdiagramme für den Maschinenzyklus M1 kennen lernen, muß der Begriff *T-Zyklus* bekannt sein. Jeder Maschinenzyklus ist wiederum in T-Zyklen unterteilt. T-Zyklen stimmen in Verhältnis 1 : 1 mit den Impulsen des Takteingangs auf Pin 6 des Z-80-Chips überein. Die höchste Taktfrequenz des Z-80 beträgt 4 MHz oder 4.000.000 Hertz, d.h. jeder T-Zyklus dauert 250 Nanosekunden. Die Zahl der T-Zyklen pro Maschinenzyklus schwankt mit der Funktion des Maschinenzyklus. Je komplizierter die Funktion ist, desto größer ist die Zahl der T-Zyklen. Ein Maschinenzyklus M1 besteht aus vier T-Zyklen, während eine Speicherablesung drei T-Zyklen hat. Der M1-Zyklus ist wegen der erforderlichen Befehlsentschlüsselung komplizierter.
Zusammenfassend läßt sich sagen, daß Befehle aus Maschienenzyklen bestehen, die sich wieder in T-Zyklen unterteilen. Jedes erste Byte eines Befehls ist ein Objektcode-Byte, so daß bei der Ausführung eines Befehls mindestens einen M1-Zyklus ausführt. Während es verschiedene Arten von Maschinenzyklen gibt, ist der T-Zyklus immer gleich. Es handelt sich dabei um einen Impuls von 250 Nanosekunden Dauer bei 4 MHz. Jeder Impuls hat eine positive Flanke (von log. 0 nach log. 1) und eine negative Flanke (von log. 1 nach log. 0).

Z-80-CPU-Zeitdiagramme

Bild 1-6 zeigt ohne genaue Zeitmarkierung das Verhältnis zwischen Adreß-, Daten- und Steuersignalen auf den Pins der Z-80-CPU bei einem M1-Zyklus. Dieses Zeitdiagramm und die Zeitdiagramme für alle anderen Z-80-Maschinenzyklen sind in dem *Z-80-CPU Technical Manual* erschienen. Wenn Sie das absolute Zeitverhältnis der Z-80-CPU-Signale untersuchen möchten, benötigen Sie dazu das Technical Manual. In dem vorliegenden Buch geht es dagegen in erster Linie um die Verwendung des Nanocomputers® mit den Schaltungen auf der Experimentierplatine.
Beim Übertragen der Z-80-CPU-Signale zur Experimentierplatine benutzten die Entwickler des Nanocomputers® eine gewöhnliche Mikro-

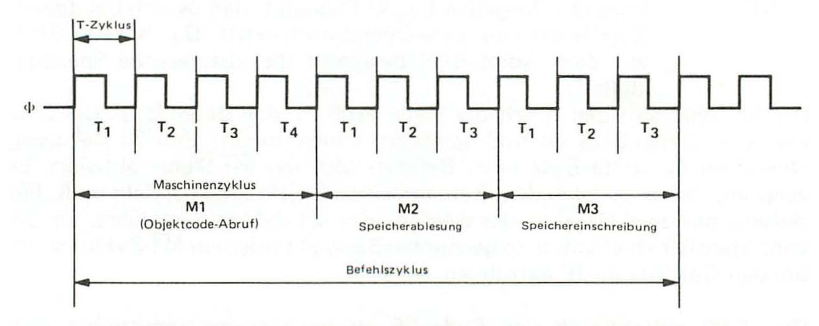


Bild 1-5. Zusammensetzung eines Z-80-Befehls aus Maschinen- und T-Zyklen

computer-BUS-Technik mit Pufferung der CPU-Signale. Die folgenden Kapitel gehen näher darauf ein. Die Pufferung ist an dieser Stelle lediglich erwähnt, weil es sich bei den erzeugten Signalen der Experimentierplatine nicht exakt um die CPU-Pinsignale handelt. Der Pufferungsvorgang und andere Schaltungen auf der Experimentierplatine des Nanocomputers® versehen die Signale auf den CPU-Pins mit verschiedenen Totzeiten (in der Größenordnung von ca. 30 bis 100 Nanosekunden). Da es Situationen gibt, wo diese Totzeiten wichtig sind, wird in der Folge zwischen den CPU- und Experimentieraufbausignalen unterschieden, indem die Letzteren mit dem Vorzeichen "B" versehen sind.
So bezieht sich A0 . . . A15 auf die CPU-Pin-Adreßleitungen, während BA0 . . . BA15 sich auf die Leitungen des Experimentieraufbaus bezieht. Für das Interfacing des Nanocomputers® ist eine genaue Zeitberechnung auf den "B"-Leitungen von größter Bedeutung. Deshalb sind neben den Z-80-CPU-Signal-Zeitdiagrammen gleichzeitig die BUS-Zeitdiagramme des Nanocomputers® abgebildet.

Bild 1-7 zeigt die genaue Zeitaufteilung für die BUS-Signale des Nanocomputers® bei einem M1-Zyklus. Es ist äußerst wichtig, Zeitdiagramme zu verstehen. Sie sind das wichtigste Hilfsmittel der Hersteller und Digital-Entwickler, um die Arbeitsweise von Chips bzw. Schaltungen zu erklären. Daher befaßt sich dieser Abschnitt ausführlich mit dem Zeitdiagramm für den M1-Zyklus des Nanocomputers®. Die gleichen Prinzipien lassen sich auch für das Lesen der in den folgenden Kapiteln behandelten Zeitdiagramme für andere Z-80-CPU- und Nanocomputer®-Maschinenzyklen anwenden.

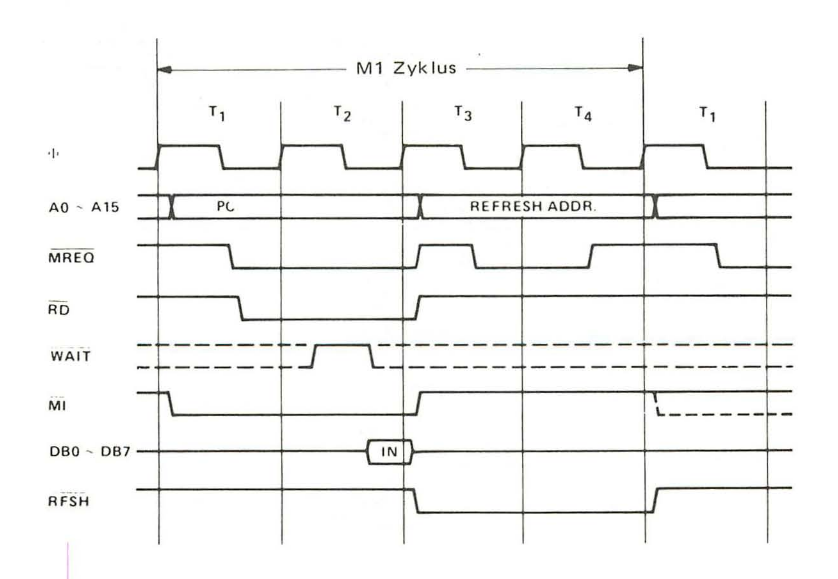


Bild 1-6. Z-80-CPU-Zeitaufteilung für einen M1-(Objektcode-Abruf)-Zyklus.

Bild 1-7 zeigt die relative Zeitaufteilung für acht bestimmte Signale: BΦ, BA0 . . . 15, $\overline{\text{BMREQ}}$, $\overline{\text{BRD}}$, $\overline{\text{BWAIT}}$ $\overline{\text{BM1}}$, BD0 . . . 7 und $\overline{\text{BRFSH}}$. Der Zeitbezug ist das BΦ-Signal, das eine graphische Darstellung des Z-80-Takteingangs ist. Die Rechteckimpulsfolge stellt logische Übergänge von low nach high und wieder nach low mit positiver Logik dar. Logisch 0 ist graphisch durch eine waagerechte Linie dargestellt, die unter der Linie für logisch 1 liegt. Die Zeit nimmt von links nach rechts zu, daher werden die T-Zyklen mit T1, T2 usw. bezeichnet, wobei T2 zeitlich hinter T1 erfolgt. Das Diagramm versucht die Realität wiederzugeben, indem es die verzögerten Übergänge von logisch 0 zu logisch 1 (oder umgekehrt) zeigt. Das ist durch die schrägen Impulsflanken dargestellt, welche die High- und Low-Zustände miteinander verbinden.

Wie im Diagramm vermerkt, hat jeder T-Zyklus eine Dauer von genau tB0 = 408 Nanosekunden. Das Taktsignal ist symmetrisch, d.h., die High-Zeit entspricht der Low-Zeit = 204 Nanosekunden (ns). Das ganze Diagramm

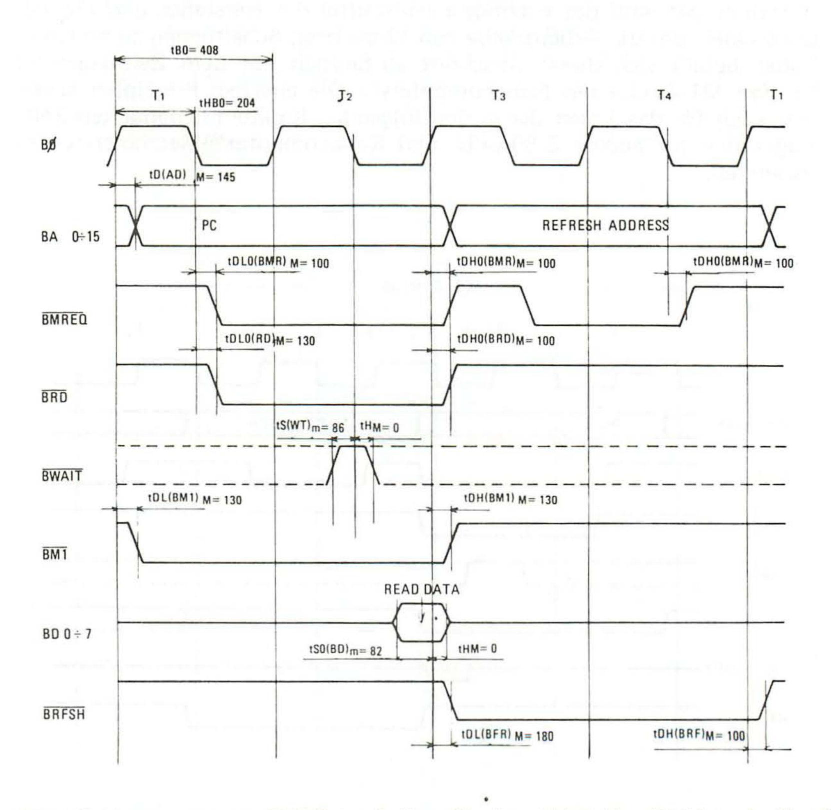


Bild 1-7. Nanocomputer®-BUS-Zeitaufteilung für einen M1-Zyklus (Objektcode-Abruf).

zeigt vier T-Zyklen, also eine Dauer von 1632 ns für die Z-80-CPU-Operation.

Die graphische Darstellung des Adreß-BUS BA0 . . . 15 ist zunächst etwas verwirrend, weil nicht eine einzige Linie mit der Darstellung von BΦ identisch ist. Das ist darauf zurückzuführen, daß die graphische Darstellung anstelle von einem Pin des Z-80-CPU-Chips insgesamt 16 Pins nämlich BA0 . . . BA15 darzustellen versucht. In einem idealen Zeitdiagramm müßte es eine hundertprozentige Übereinstimmung zwischen Signalen und Kurven geben, wenn man einmal von den Adreß- und Datenpins absieht, wo dies fast immer der Fall ist. In der Darstellung Bild 1-7 sind die Adressen und Daten jedoch angeglichen. Das hat folgende Gründe:

a) Es ist sehr schwierig, jedes Adreß- oder Datensignal in einer graphischen Darstellung separat aufzuzeichnen.
b) Es ist einfacher und meistens aufschlußreicher, die in Bild 1-7 gezeigte Doppellinie zu zeichnen und die in ihnen enthaltenen Adressen oder Daten bzw. Zeiten anzugeben.
c) Die Schnittpunkte, an denen sich die Linien kreuzen, stellen die Zeit für die in den Adreß- oder Datenlinien enthaltenen Datenangaben dar.

Daher geht aus dem Zeitdiagram in Bild 1-7 hervor, daß die 16 Adreßpins den Inhalt des PC-Registers und schließlich eine "Refesh-Adresse" bei einem M1-Zyklus enthalten. Weiter wird angegeben, daß $tD(AD)_M = 145$ ist, was bedeutet, daß die zwischen der ansteigenden Flanke von T1 (dabei wird BΦ logisch 1) bis zur Auflage des PC auf den Adreß-BUS verstrichene Zeit nicht länger als 145 ns ist. Bei den feineren senkrechten Linien in dem Diagramm handelt es sich lediglich um Hilfslinien.

Was den Inhalt des auf die Adreßleitungen gelegten PC anbetrifft, so käme eine interessantere Zeit als die angegebene heraus, wenn sich das PC auf den Adreßleitungen *stabilisiert* hat. Diese Stabilisierung von Daten tritt immer ein paar Nanosekunden *nach* dem ersten Auflegen der Information auf die Leitungen ein. Deshalb ist es gewöhnlich weniger interessant zu erfahren, wann die Daten zuerst auf die Daten- oder Adreßleitungen gelangt sind als festzustellen, wann sich die Information dort stabilisiert hat. Auch die genaue Zeit für den Austausch des PC-Inhalts mit der "Refresh-Adresse" ist in Bild 1-7 nicht angegeben.

Ein Hauptzweck der Zeitdiagramme ist die Darstellung des relativen Signalverhaltens in einer bestimmten Operation. Man beachte, daß ohne die *absoluten* Zeitangaben lediglich relative Zeitinformation im Diagramm von Bild 1-7 enthalten ist. Häufig ist es ausreichend, nach der Relativzeit zu arbeiten, die sich immer auf den Systemtakt bezieht, nämlich im genannten Beispiel das BΦ-Signal. Nur selten fehlt in einem Zeitdiagramm der Bezugstakt. Es folgen nun einige relative Zeitbeobachtungen beim einem Z-80-CPU-M1-Zyklus.
Direkt hinter der ansteigenden Flanke von T1 wird das $\overline{BM1}$-Signal aktiviert. Das heißt, das $\overline{BM1}$-Signal geht aus dem High- in den Low-Zustand über (blockierte Signale sind normalerweise high, während unblockierte

Signale normalerweise low sind). Auch der Programmzähler bzw. der Inhalt des PC-Registers wird auf die 16 Adreßleitungen BA0 . . . BA15 gegeben (die in der Folge häufig als Adreß-BUS bezeichnet sind). Die abfallende (negative Flanke des T1 aktiviert die $\overline{BMREQ}$- und $\overline{BRD}$-Signale. Einen kompletten T-Zyklus später soll der Speicher durch Auflegen des spezifischen Speicherplatzinhaltes auf die Datenleitungen BD0 . . . BD7 (Daten-BUS) ansprechen. Das heißt: Die Speicher-ICs und ihre Schaltungen haben ca. einen T-Zyklus lang Zeit,um festzustellen, welcher spezielle Speicherplatz angesprochen ist. Der Zugriff besteht aus acht Datenbits. Je kürzer der T-Zyklus ist, desto zwingender sind Zeitansprüche an den Speicher. Selbstverständlich muß man für die Leistung immer bezahlen, daher ist ein schnellerer Speicher teuerer als ein langsamer. Das $\overline{BM1}$-Signal wird hinter der ansteigenden (positven) Flanke von T3 deaktiviert (auf high gebracht). Zur gleichen Zeit hat die CPU die Speicherdaten abgelesen, so daß man sie aus dem Daten-BUS nehmen kann. Sobald die Speicher-Ableseoperation beendet ist, beginnt ein neuer Zyklus (der noch Bestandteil desselben M1-Zyklus ist). Es ist die *Refresh-Operation.* Dies ergibt sich daraus, daß das $\overline{BRFSH}$-Signal nicht eher aktiviert wird, bis die ansteigende Flanke von T3 gleichzeitig mit der Refresh-Adresse auf dem Adreß-BUS ist.
Auf die Speicher-Refresh-Operation wird an dieser Stelle nicht eingegegangen; sie findet ausschließlich während der M1-Zyklen statt.

Ein Signal ist bisher noch nicht erwähnt, nämlich das $\overline{BWAIT}$-Signal. Der $\overline{BWAIT}$-Anschluß des Nanocomputers® ist direkt mit dem $\overline{WAIT}$-Pin (Eingangspin 24) der Z-80-CPU verbunden. Es handelt sich dabei um einen Eingabepin, den die CPU nur zeitweilig abtastet. Die andere Zeit über spielt der logische Zustand des $\overline{WAIT}$-Pins keine Rolle; die gestrichelte Linie stellt diese Zeitpunkte graphisch dar. Bei einem M1-Zyklus tastet die CPU den $\overline{WAIT}$-Pin nur an der abfallenden Flanke des Taktes bei T2 ab. Das Signal ist inaktiv, wenn der logische Zustand high ist. Die CPU führt dann die normalen M1-Zyklus-Operationen durch. Ist das $\overline{BWAIT}$-Signal jedoch low (aktiv), dann ist ein Wartezustand bei dem M1-Zyklus erforderlich. Wartezustände werden dazu benutzt, um die für eine Speicherantwort auf eine Zugriffanforderung zur Verfügung stehende Zeit über einen T-Zuklus hinaus zu verlängern. Bei jedem zusätzlichen Wartezustand steht dem Speicher für seine Antwort ein zusätzlicher T-Zyklus zur Verfügung. Während der abfallenden Flanke des T-Zyklus taktet die CPU den $\overline{WAIT}$-Pin erneut ab, um festzustellen, ob er den inaktiven Zustand erreicht hat. Ist dies nicht der Fall, fügt die CPU einen weiteren Wartezustand hinzu. Im positiven Falle setzt die CPU die Ablesung der Daten vom Daten-BUS sowie die Refresh-Operation fort. Bild 1-8 zeigt die Zeitaufteilung für einen M1-Zyklus mit zwei eingebauten Wartezuständen bei einem Nanocomputer®-BUS.

Das $\overline{BWAIT}$-Signal ist während zwei abfallender Flanken des BΦ-Signals logisch 0 und ist dann für logisch 1 freigegeben. Während der restlichen Zeit ist der Zustand des $\overline{BWAIT}$-Signals irrelevant, weil es von der CPU nicht abgetastet wird.
Bild 1-9 zeigt das Zeitdiagramm für einen Ausgabezyklus an ein externes

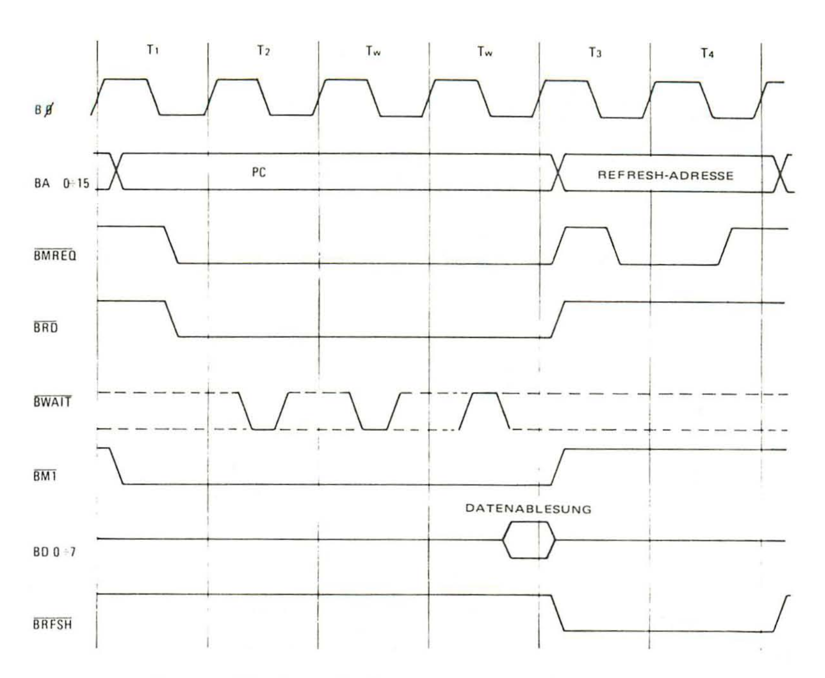


Bild 1-8. Zeitaufteilung für einen M1-Zyklus mit zwei eingebauten Wartezuständen einem Nanocomputer®-BUS.

Gerät. Es sei angenommen, daß die Ausgabe-Anweisung (01H),A (hex D3 01) erfolgt. Diese Anweisung befiehlt, den Inhalt des Akkumulators an den Ein-/Ausgabebaustein 01 abzugeben.
Auf diese Anweisung wird in den nächsten Kapiteln näher eingegangen. Nach Ausführung des M1-Zyklus zum Ablesen des D3-Objektcodes vom Speicher bzw. zu seiner Entschlüsselung weiß die CPU, daß ein 2-Byte-Ausgabebefehl vorliegt, dessen zweites Byte die Zahl des Ein-/Ausgabebausteins ist. Es wird ein Speicherzyklus zur Ablesung der nächsten Speicherstelle und Eingabe des 01 in die CPU eingeleitet. Der nächste von der CPU ausgeführte Zyklus ist der in Bild 1-8 dargestellte Ausgabezyklus. Man beachte, daß die beteiligten Steuersignale $\overline{BWR}$ und $\overline{BIORQ}$ sind. Die Adresse 01 des Ein-/Ausgabebausteins (auch periphere Adresse genannt), wird auf die niederwertigen acht Bits des Adreß-BUS BA0 . . . 7 gelegt, während der Inhalt des Akkumulators den Daten-BUS BD0 . . . 7 belegt. Sowohl der Daten- als auch der Adreß-BUS enthalten die richtige Information, wenn die beiden Steuersignale $\overline{BWR}$ und $\overline{BIORQ}$ aktiviert werden. Das ist kritisch, weil das Ein-/Ausgabegerät die gleichzeitige Aktivierung der $\overline{BWR}$ und $\overline{BIORQ}$ und seine Zahl 01 auf dem Adres-BUS abtastet und die Information sofort aus dem Daten-BUS herausnimmt. Beachten Sie, daß der Wartezustand Tw* automatisch erscheint, denn er wird nicht von dem $\overline{BWAIT}$-Signal angefordert ($\overline{BWAIT}$ ist während der gesamten Zeit high). Die zusätzliche Tw*-Periode gibt der peripheren Schaltung ein Bit mehr Zeit zum Antworten.

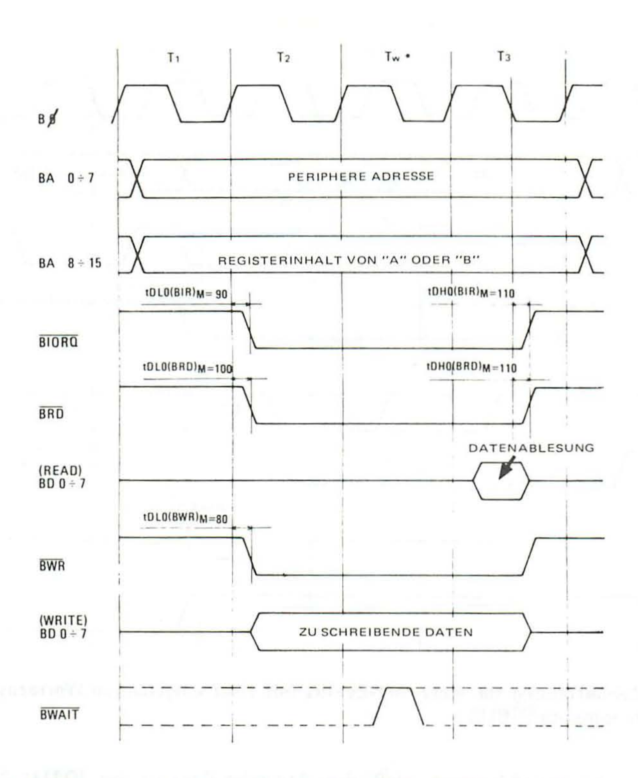


Bild 1-9. Zeiteinteilung für einen Ausgabezyklus beim Nanocomputer®

Der Vollständigkeit halber wird die absolute Zeitaufteilung für die Z-80-CPU-Signale in Bild 1-10 dargestellt. Für den Entwurf einer Schaltung mit der Z-80-CPU (wie z.B. der Nanocomputer®) liefert dieses Diagramm wichtige Aufschlüsse. Tabelle 1-1 gibt einen Querverweis zwischen den in Bild 1-10 dargestellten Zeitintervallen und den verstrichenen Nanosekunden mit einer Taktfrequenz von 2,5 MHz. Die beschriebenen Grundprinzipien zum Lesen eines Zeitdiagramms treffen auch für Bild 1-10 zu. Da auf alle in diesem Diagramm vorkommenden Signale in den folgenden Kapiteln noch näher eingegangen wird, ist an dieser Stelle keine weitere Erklärung notwendig.

Damit ist die Kurz-Einführung in das Z-80-Interfacing abgeschlossen. Die Grundelemente sind in den behandelten Themen nur kurz angerissen; es folgt in den nächsten Kapiteln eine intensivere Behandlung. Das dabei erworbene theoretische Wissen erfährt im praktischen Experiment seine Bestätigung.

Tabelle 1-1. Signalzeiten der Z-80-CPU

$T_A = 0°C$ to $70°C$, $V_{cc} = +5V \pm 5\%$, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	.4	[12]	µsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	180	[E]	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	$C_L = 50pF$
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		230	nsec	$C_L = 50pF$
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		100	nsec	$C_L = 50pF$
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		90	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		110	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		110	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		100	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		130	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		100	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		110	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		80	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		90	nsec	
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		130	nsec	$C_L = 50pF$
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		130	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		180	nsec	$C_L = 50pF$
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		150	nsec	
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	$C_L = 50pF$
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		120	nsec	$C_L = 50pF$
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		110	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 210$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$

[8] $t_{w(MRL)} = t_c - 40$

[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$

[10] $t_{w(\overline{WR}L)} = t_c - 40$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 80$

NOTES

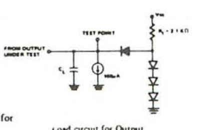
A. Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

D. Output Delay vs. Loaded Capacitance
   TA = 70°C   Vcc = +5V ±5%
   Add 10nsec delay for each 50pf increase in load up to a maximum of 200pf for the data bus & 100pf for address & control lines

E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 µsec maximum



Load circuit for Output

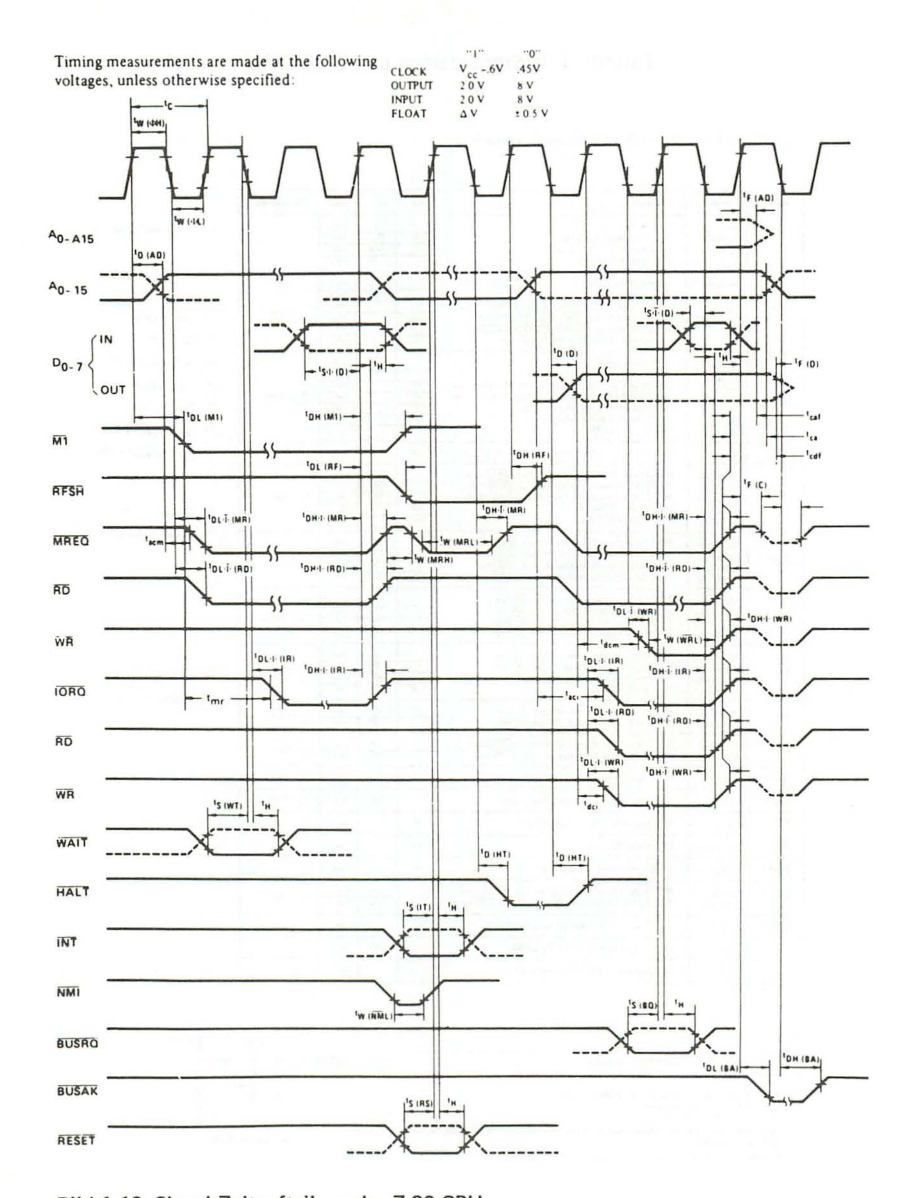


Bild 1-10. Signal-Zeitaufteilung der Z-80-CPU

Kapitel 2

Experimentier-Platine NEZ80

Dieses Kapitel macht Sie mit der Experimentier-Platine bekannt, die Sie zusammen mit Ihrem Nanocomputer® für Mikrocomputer-Interfacing-Experimente benutzen können. Ein kompletter Satz von schematischen Darstellungen vervollständigt Ihre Kenntnisse über die genauen Schaltungen für LED-Monitoren, entprellte Logikschalter und Impulsgeber sowie andere zur Platine gehörende Bauelemente.

Nach dem Studium dieses Kapitels werden Sie folgendes können:

- die schematischen Darstellungen der Experimentier-Platine NEZ80 lesen und verstehen;
- einfache Versuche mit Hilfe der Schalter, LED-Monitoren und Impulsgeber durchführen;
- etwa die Hälfte der zu den 40-Pin-Anschlüssen an der Experimentier-Platine geleiteten Signale kurz erklären. Die übrigen Signale werden in den folgenden Kapiteln eingehend behandelt.

Bild 2-1 zeigt die Draufsicht auf die Experimentier-Platine NEZ80. Die Hauptbestandteile sind:

Anschlüsse J1, J2 und J3
1 40-Pin-PIO-Anschluß
1 Lochrasterplatte SK-10
40-Pin-DIP-Anschlüsse A, B und C
8 LED-Monitoren, gekennzeichnet von rechts nach links mit LM0, LM1, LM2, LM3, LM4, LM5, LM6 und LM7
8 Logikschalter, von rechts nach links gekennzeichnet mit SW0, SW1, SW2, SW3, SW4, SW5, SW6 und SW7
2 Impulsgeber, von rechts nach links mit P0 und P1 gekennzeichnet
1 LED-Stromanzeige mit +5V gekennzeichnet

In den folgenden Abschnitten werden alle genannten Bestandteile der Platine näher beschrieben.

Anschlüsse J1, J2 und J3

Die Anschlüsse J1 und J2 sind wesentliche Bestandteile der Interface-Schaltung zwischen der CPU des Nanocomputers® und der Experimentierplatine. Mit Hilfe dieser Anschlüsse können Sie Signale zwischen der

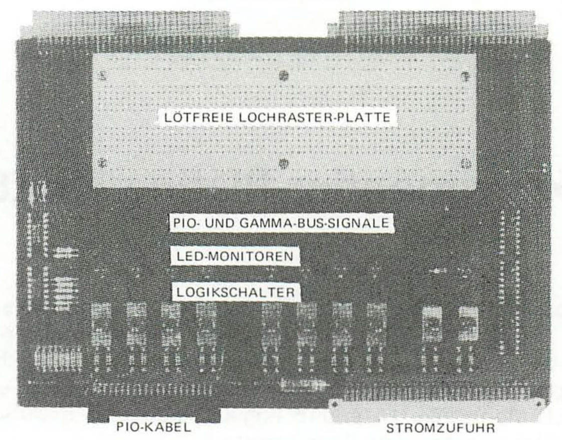


Bild 2-1. Experimentier-Platine NEZ80.

CPU und der von Ihnen auf der Lochrasterplatte SK-10 aufgebauten Schaltung austauschen. Die Pinbelegung der Anschlüsse J1 und J2 zeigt Bild 2-2. Stören Sie sich an dieser Stelle nicht an die noch unbekannten Signalbezeichnungen; auf die gehen spätere Abschnitte ein.

Der Anschluß J3 dient der Stromversorgung der Experimentierplatine. Bild 2-3 zeigt die schematische Darstellung des Anschlusses J3 für die Stromversorgung +5V, –5V, +12V und –12V.

Der PIO-Anschluß

Die CPU-Platine des Nanocomputers® hat zwei ICs für parallele Ein- und Ausgabe: die Z-80-PIOs. PIO 1 stellt die Verbindung zwischen dem Tastenfeld und dem Display des Nanocomputers® her. PIO 2 steht jedem informierten Benutzer zur Verfügung. Zur Benutzung von PIO 2 werden zwei Signale benötigt. Der PIO-Anschluß verbindet die Experimentier-Platine über ein 40-adriges elastisches Kabel mit dem PIO 2, der verschiedene Anschlußstifte mit der Lochrasterplatte SK-10 koppelt.
Die Anschlußbelegung des PIO-Anschlusses, der die Verbindung mit dem Anschluß J7 auf der CPU-Platine (NBZ80) herstellt, zeigt Bild 2-4. Die zum PIO-Anschluß gehörenden Signale wie PCn, PDn, CRDY, $\overline{\text{CSTB}}$, usw. sind in dem Abschnitt über die Anschlüsse A, B und C beschrieben.

Die Lochrasterplatte SK-10

Die Lochrasterplatte auf der Experimentierplatine NEZ80 kann man in passender Form auch im Fachhandel beschaffen. Die Lochrasterplatte ist zur Aufname von ICs, Widerständen, Kondensatoren und allen anderen Bauelementen, die Sie für die in diesem Buch beschriebenen Versuche benötigen, konzipiert.
Bild 2-5 zeigt die Draufsicht der Lochrasterplatte SK-10. Sie verfügt über

Pin	J1 a	J1 c	J2 a	J2 c
1	+5V	+5V	+5V	+5V
2		BA3		
3		BAØ	−12V	−12V
4		BA4	−5V	−5V
5		BA5		$\overline{\text{IOQ3}}$
6		BA2		$\overline{\text{IOEØ}}$
7		BA1		BA13
8		BF CU		$\overline{\text{IOE1}}$
9				$\overline{\text{IOE2}}$
10				BD7
11		$\overline{\text{BRFSH}}$		$\overline{\text{IOE3}}$
12		BA1Ø		BD6
13	*IEI	IEO		$\overline{\text{BRD}}$
14	*DEI	DEO*		$\overline{\text{BWR}}$
15		$\overline{\text{BM1}}$		$\overline{\text{BDOUT}}$*
16	+12V	+12V	+12V	+12V
17		BØ		BA15
18		$\overline{\text{BHALT}}$		BA14
19				BA12
20		$\overline{\text{BBUSAK}}$		$\overline{\text{DBIN}}$*
21		$\overline{\text{IOU3}}$		$\overline{\text{BIORQ}}$
22		$\overline{\text{IOU2}}$		$\overline{\text{BMREQ}}$
23		$\overline{\text{IOU1}}$		$\overline{\text{BNMI}}$
24		$\overline{\text{IOUØ}}$		$\overline{\text{BINT}}$
25		BD2		$\overline{\text{BBUSRQ}}$
26		BD1		$\overline{\text{BWAIT}}$
27		BDØ		BA11
28		$\overline{\text{BRESET}}$		BA9
29		BD3		BA7
30		BD4		BA6
31		BD5		BA8
32	GND	GND	GND	GND

$\overline{\text{DBIN}}$ und $\overline{\text{DBOUT}}$ werden nur zu Versuchszwecken benutzt

* Anschlüsse sind nicht mit der Lochrasterplatte verbunden.

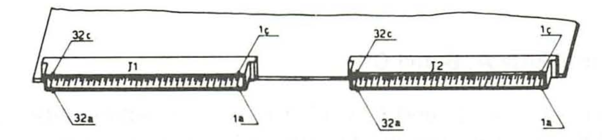


Bild 2-2.Anschlüsse J1 und J2.

64 × 2 Sätze von je fünf elektrisch verbundenen lötfreien Anschlüssen, die sich beiderseits einer schmalen Mittelrille erstrecken, sowie acht Sätze von 25 elektrisch verbundenen Anschlüssen an den Rändern. Die mittleren Gruppen von fünf elektrisch verbundenen Anschlüssen nehmen integrierte Schaltungen auf (mit 14 und 16 Anschlußstiften). Die Steckverbindungen ermöglichen bis zu vier zusätzliche Anschlüsse an jedem Pin der ICs. Die Gruppen mit 25 elektrisch verbundenen Anschlüssen an den Rändern sind zur Versorgung der integrierten Schaltungen und der anderen Bauelemente bestimmt. Die Unterseite der Lochrasterplatte SK-10 – nach Entfernung des Schutzmaterials – wird in Bild 2-6 gezeigt. Aus dem Foto geht deutlich hervor, welche lötfreien Anschlüsse elektrisch leitend verbunden sind.

Die SK-10 besteht aus einem präzisionsgeformten, schlagfesten Kunststoffgehäuse mit insgesamt 840 korrosionsfreien Federkontakten und einer Lebenserwartung von über 10 000 Drahteinführungen. Die lötfreien Anschlüsse nehmen Drahtstärken in gebräuchlicher Stärke auf.

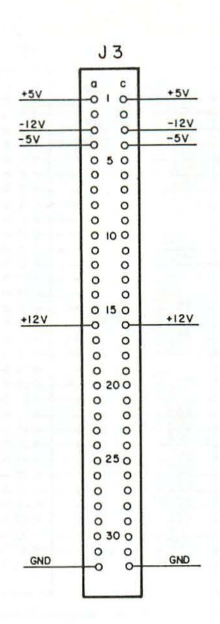


Bild 2-3. Pinbelegung beim Anschluß J3.

40-Pin-DIP-Anschlüsse A, B und C

Bei den Anschlüssen A, B und C handelt es sich um 40-Pin-DIP (DUAL-IN-LINE-PACKAGE)-Gehäuse in Standardausführung, die über die Experimentier-Platinenschaltung mit den Anschlüssen J1, J2 und PIO verbunden sind. Insgesamt werden 86 Signale zu den Anschlüssen A, B und C geleitet.

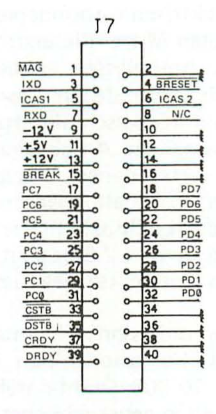


Bild 2-4. PIO-Anschluß auf der NEZ80-Platine.
Die mit der Lochrasterplatte verbundenen Anschlüsse sind PC0 . . . PC7, PD0 . . . PD7, $\overline{\text{CSTB}}$, $\overline{\text{DSTB}}$, CRDY und DRDY.

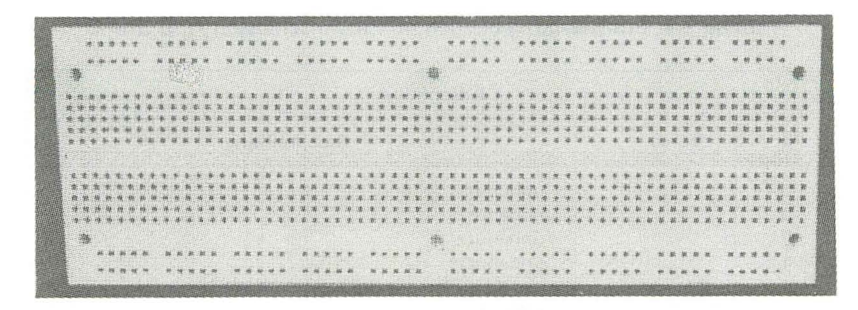


Bild 2-5. Draufsicht der Lochrasterplatte SK-10.

Diese Signale stehen einer Schaltung auf der Lochrasterplatte SK-10 durch bloßes Installieren von Drähten zwischen der SK-10-Schaltung und den entsprechenden Löchern der Anschlüsse A, B und C zur Verfügung. Die an den Anschlüssen A, B und C anstehenden Signale sind in Tabelle 2-1 aufgeführt.

Z-80-CPU (gepufferter BUS): 38 Signale = eins pro Pin außer für V_{CC} ($+U_b$) und GND + 1 für einen verzögerten $\overline{BWR}$-Impuls.

PIO 2: 21 Signale für zwei Daten-BUS-Leitungen und die dazugehörigen Steuersignale plus ein Prioritäts-Interrupt-Signal (IEQ).

LED-Monitoren: 8 LED-Treiberausgänge, ein Ausgang pro Bit in einem 8-Bit-Byte.

Impulsgeber: 4 entprellte Signale, eins pro Wert des Impulsgebers sowie seines logischen Komplements.

I/O-Steuerung: 9 Signale, sie entschlüsseln teilweise die niederwertigen acht Bits des Adreß-BUSSES.

Stromverzorgung: +5V, –5V, +12V und –12V mit GND.

In Tabelle 2-1 ist die genaue Pin-Belegung für jedes der 40-Pin-Anschlüsse A, B und C mit einer Beschreibung für jedes Signal angegeben.

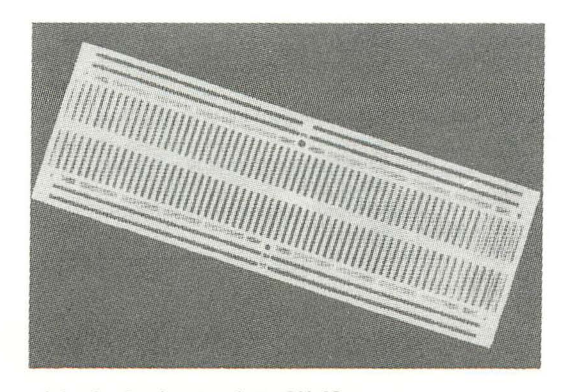


Bild 2-6. Unteransicht der Lochrasterplatte SK-10.

Tabelle 2-1. Pin-Belegung bei den 40-Pin-Anschlüssen A, B und C

Anschluß A		
Pin	**Signal**	**Beschreibung**
1		unbenutzt
2		unbenutzt
3		unbenutzt
4		unbenutzt
5	$\overline{P1}$	das Komplement des logischen Wertes von Impulsgeber 1
6	$\overline{P0}$	der logische Wert des Impulsgebers 1
7		unbenutzt
8	P0	das Komplement des logischen Wertes von Impulsgeber 0
9	P0	der logische Wert des Impulsgebers 0
10		unbenutzt
11		unbenutzt
12	SW7	Logikschalter 7
13	SW6	Logikschalter 6
14	SW5	Logikschalter 5
15	SW4	Logikschalter 4
16	SW3	Logikschalter 3
17	SW2	Logikschalter 2
18	SW1	Logikschalter 1
19	SW0	Logikschalter 0
20		unbenutzt
21		unbenutzt
22	LM0	LED-Monitor 0
23	LM1	LED-Monitor 1
24	LM2	LED-Monitor 2
25	LM3	LED-Monitor 3
26	LM4	LED-Monitor 4
27	LM5	LED-Monitor 5
28	LM6	LED-Monitor 6
29	LM7	LED-Monitor 7
30		unbenutzt
31		unbenutzt
32	GND	Masse
33	−12V	Versorgungsspannung −12 Volt
34	GND	Masse
35	+12V	Versorgungsspannung +12 Volt
36	GND	Masse
37	−5V	Versorgungsspannung −5 Volt
38	GND	Masse
39	+5V	Versorgungsspannung +5 Volt
40	GND	Masse

Tabelle 2-1. Pin-Belegung bei den 40-Pin-Anschlüssen A, B und C

Anschluß B		
Pin	**Signal**	**Beschreibung**
1	$\overline{BM1}$	gepufferter $\overline{M1}$-Signal-Ausgang der Z-80-CPU
2	$\overline{BMREQ}$	gepufferter $\overline{MREQ}$-Signal-Ausgang der Z-80-CPU
3	$\overline{BIORQ}$	gepufferter $\overline{IORQ}$-Signal-Ausgang der Z-80-CPU
4	BRFSH	gepufferter RFSH-Signal-Ausgang der Z-80-CPU
5		unbenutzt
6	$\overline{BRD}$	gepufferter $\overline{RD}$-Signal-Ausgang der Z-80-CPU
7	$\overline{BWR}$	gepufferter $\overline{WR}$-Signal-Ausgang der Z-80-CPU
8	DBWR	gepufferter $\overline{BWR}$-Signal-Ausgang der Z-80-CPU
9		unbenutzt
10	$\overline{BHALT}$	gepufferter $\overline{HALT}$-Signal-Ausgang der Z-80-CPU
11	$\overline{BWAIT}$	gepufferter $\overline{WAIT}$-Signal-Eingang der Z-80-CPU
12	$\overline{BINT}$	gepufferter $\overline{INT}$-Signal-Eingang der Z-80-CPU
13	$\overline{BNMI}$	gepufferter $\overline{NMI}$-Signal-Eingang der Z-80-CPU
14	$\overline{BRESET}$	gepufferter $\overline{RESET}$-Signal-Eingang der Z-80-CPU (muß offener Kollektor-Ausgang von der Anwender-Schaltung sein)
15	unbenutzt	
16	$\overline{BBUSRQ}$	gepuffeter $\overline{BUSRQ}$-Signal-Eingang vom Z-80-CPU-Chip (muß offener Kollektor-Ausgang der Anwender-Schaltung sein)
17	$\overline{BBUSAK}$	gepufferter $\overline{BUSAK}$-Signal-Ausgang der Z-80-CPU
18		unbenutzt
19	BΦ	gepufferter Z-80-Takt
20		unbenutzt
21		unbenutzt
22	BDO	gepufferte DO-Leitung des bidirektionalen Z-80-Daten-BUSSES
23	BD1	gepufferte D1-Leitung des bidirektionalen Z-80-Daten-BUSSES
24	BD2	gepufferte D2-Leitung des bidirektionalen Z-80-Daten-BUSSES
25	BD3	gepufferte D3-Leitung des bidirektionalen Z-80-Daten-BUSSES
26	BD4	gepufferte D4-Leitung des bidrektionalen Z-80-Daten-BUSSES
27	BD5	gepufferte D5-Leitung des bidirektionalen Z-80-Daten-BUSSES
28	BD6	gepufferte D6-Leitung des bidirektionalen Z-80-Daten-BUSSES
29	BD7	gepufferte D7-Leitung des bidirektionalen Z-80-Daten-BUSSES
30		unbenutzt
31	$\overline{IOQ3}$	erzeugter Negativ-Impuls, wenn A7-A0 = 0000 11xx
32	$\overline{IOE0}$	erzeugter Negativ-Impuls, wenn A7-A0 = 0001 00xx
33	$\overline{IOE1}$	erzeugter Negativ-Impuls, wenn A7-A0 = 0001 01xx
34	$\overline{IOE2}$	erzeugter Negativ-Impuls, wenn A7-A0 = 0001 10xx
35	$\overline{IOE3}$	erzeugter Negativ-Impuls, wenn A7-A0 = 0001 11xx
36	$\overline{IOUO}$	erzeugter Negativ-Impuls, wenn A1-A0 = 00 und BIORQ aktiv ist (logisch 0)

Tabelle 2-1. Pin-Belegung bei den 40-Pin-Anschlüssen A, B und C

37	$\overline{IOU1}$	erzeugter Negativ-Impuls, wenn A1-A0 = 01 und BIORQ aktiv ist (logisch 0)
38	$\overline{IOU2}$	erzeugter Negativ-Impuls, wenn A1-A0 = 10 und BIORQ aktiv ist (logisch 0)
39	$\overline{IOU3}$	erzeugter Negativ-Impuls, wenn A1-A0 = 11 und BIORQ aktiv ist (logisch 0)
40		unbenutzt

Anschluß C		
Pin	**Signal**	**Beschreibung**
1	CRDY	Bereitschaftssignal für Gatter A vom PIO 2
2	$\overline{CSTB}$	Abtastsignal-Eingabe an PIO 2 von Gatter A
3	PC7	D7-Leitung des bidirektionalen Daten-BUS von Gatter A für PIO 2
4	PC6	D6-Leitung des bidirektionalen Daten-BUS von Gatter A für PIO 2
5	PC5	D5-Leitung des bidirektionalen Daten-BUS von Gatter A für PIO 2
6	PC4	D4-Leitung des bidirektionalen Daten-BUS von Gatter A für PIO 2
7	PC3	D3-Leitung des bidirektionalen Daten-BUS von Gatter A für PIO 2
8	PC2	D2-Leitung des bidirektionalen Daten-BUS von Gatter A für PIO 2
9	PC1	D1-Leitung des bidirektionalen Daten-BUS von Gatter A für PIO 2
10	PC0	D0-Leitung des bidirektionalen Daten-BUS von Gatter A für PIO 2
11	DRDY	Bereitschaftssignal für Gatter B von PIO 2
12	$\overline{DSTB}$	Abtastsignal-Eingabe an PIO 2 von Gatter B
13	PD7	D7-Leitung des bidirektionalen Daten-BUS von Gatter B für PIO 2
14	PD6	D6-Leitung des bidirektionalen Daten-BUS von Gatter B für PIO 2
15	PD5	D5-Leitung des bidirektionalen Daten-BUS von Gatter B für PIO 2
16	PD4	D4-Leitung des bidirektionalen Daten-BUS von Gatter B für PIO 2
17	PD3	D3-Leitung des bidirektionalen Daten-BUS von Gatter B für PIO 2
18	PD2	D2-Leitung des bidirektionalen Daten-BUS von Gatter B für PIO 2
19	PD1	D1-Leitung des bidirektionalen Daten-BUS von Gatter B für PIO 2

Tabelle 2-1. Pin-Belegung bei den 40-Pin-Anschlüssen A, B und C

20	PD0	D0-Leitung des bidirektionalen Daten-BUS von Gatter B für PIO 2
21		unbenutzt
22	BA0	gepufferte A0-Leitung des Z-80 Adreß-BUS
23	BA1	gepufferte A1-Leitung des Z-80 Adreß-BUS
24	BA2	gepufferte A2-Leitung des Z-80 Adreß-BUS
25	BA3	gepufferte A3-Leitung des Z-80 Adreß-BUS
26	BA4	gepufferte A4-Leitung des Z-80 Adreß-BUS
27	BA5	gepufferte A5-Leitung des Z-80 Adreß-BUS
28	BA6	gepufferte A6-Leitung des Z-80 Adreß-BUS
29	BA7	gepufferte A7-Leitung des Z-80 Adreß-BUS
30		unbenutzt
31	BA8	gepufferte A8-Leitung des Z-80 Adreß-BUS
32	BA9	gepufferte A9-Leitung des Z-80 Adreß-BUS
33	BA10	gepufferte A10-Leitung des Z-80 Adreß-BUS
34	BA11	gepufferte A11-Leitung des Z-80 Adreß-BUS
35	BA12	gepufferte A12-Leitung des Z-80 Adreß-BUS
36	BA13	gepufferte A13-Leitung des Z-80 Adreß-BUS
37	BA14	gepufferte A14-Leitung des Z-80 Adreß-BUS
38	BA15	gepufferte A15-Leitung des Z-80 Adreß-BUS
39		unbenutzt
40		unbenutzt

PIO 2 hat folgende Adressen: Gatter C (Daten) = 08H, Gatter C (Steuerung) = 0AH, Gatter D (Daten) = 09, Gatter D (Steuerung) = 0BH

Bereits in Kapitel 1 ist die Pinbelegung der Z-80-CPU in Bild 1-4 dargestellt. Insgesamt 38 der Z-80-Signale gelangen zur Experimentierplatine. Folglich sind die Bezeichnungen ähnlich wie bei den Z-80-Pins mit Ausnahme des vorgesetzten Buchstaben "B", wie z.B. BD0 . . . BD7, BA0 . . . BA15, $\overline{\text{BMREQ}}$, $\overline{\text{BRD}}$, $\overline{\text{BWR}}$. Das "B" steht für gepuffert (buffered); dabei ist das Loch der Rasterplatte mit dem entsprechenden Anschlußstift der Z-80-CPU über einen Puffer verbunden (Bild 2-7).
Die Pufferung schützt die Signale auf dem Weg zu und von der Z-80-CPU. Selbst dem vorsichtigsten Benutzer können bei Schaltungen auf der Lochrasterplatte SK-10, die über die Anschlüsse A, B und C mit der CPU verbunden ist, Fehler unterlaufen. Durch das Puffern der CPU-Signale entsteht ein sinnvoller Fehlerspielraum; er verhindert, daß so manches IC beschädigt wird. Die Pufferstufen zwischen der Z-80-CPU und der Experimentierplatine schützen nicht nur die Signale. Sie dienen gleichzeitig als Treiberstufen für je 15 TTL- oder 60 Low-Power-Schottky-Bauelemente. Die Pufferung verändert nicht den logischen Zustand der entsprechenden Signale. Allerdings treten Verzögerungen der Signal-Laufzeiten ein, so daß sich das zeitliche Zusammenwirken der Signale ändern kann.
Wie aus den folgenden Kapiteln hervorgeht, ist die Unterscheidung zwischen den Z-80-CPU-Signalen und ihren dazugehörigen gepufferten

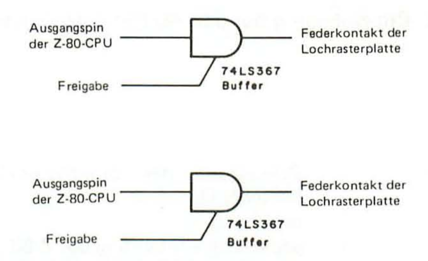


Bild 2-7. Die Federkontakte der Lochrasterplatte sind über das Buffer-IC 74LS367 mit den entsprechenden Pins der Z-80-CPU verbunden.

Signalen auf dem BUS wichtig. Es ist daher besondere Sorgfalt auf die Einhaltung der folgenden Begiffe zu verwenden:

> CPU-Signalen fehlt stets der Anfangsbuchstabe "B". Während für Abtast-Schaltungen ähnlich denen beim Nanocomputer® ungepufferte CPU-Signale Verwendung finden, dienen sie nur als Beispiel für eine Funktion, wenn die CPU-Signale auf dem BUS zur Verfügung stünden. Dies ist natürlich weder beim Nanocomputer® noch bei den meisten anderen Mikrocomputersystemen der Fall. Wann immer in einem Versuch eine graphische Darstellung erscheint, werden die BUS-Signale benutzt und mit dem Anfangsbuchstaben "B" versehen. Bei allen Verdrahtungsplänen sind die Bezeichnungen mit denen auf der Lochrasterplatte identisch.

Die meisten der erwähnten Signale entstehen auf den Anschlüssen A, B und C durch direkte Verbindung mit den Anschlußstiften an J1, J2 oder dem PIO-Anschluß. Eine Ausnahme davon bildet das $\overline{\text{DBWR}}$-Signal, bei dem es sich um das $\overline{\text{BWR}}$-Signal mit 100 Nanosekunden Verzögerung handelt. Die zur Erzeugung des $\overline{\text{DBWR}}$-Signals erforderliche Schaltung ist in Bild 2-8 dargestellt.
Viele der in Tabelle 2-1 aufgeführten Signale sind Ihnen sicherlich noch unbekannt. Die Tabellen dienen in erster Linie zu Nachschlagzwecken. Sie geben einen Überblick über die Art und die Anzahl der Signale, die für das Mikrocomputer-Interfacing zur Verfügung stehen. Die Versuche in diesem Kapitel machen Sie mit den LED-Monitoren, den Schaltern, den Impulsgebern sowie der +5V–/GND-Stromversorgung bekannt.
Achtung: Die Anschlüsse A, B und C sind mit der Oberseite der Experimentierplatine verlötet. Es ist daher sehr umständlich, die genannten Anschlüsse auszuwechseln. Unterlassen Sie deshalb alles, was die Lebensdauer der Anschlüsse stark vermindert. Es ist empfehlenswert, immer die

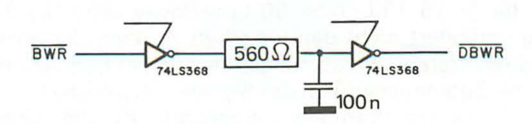


Bild 2-8. $\overline{\text{DBWR}}$-Schaltung.

drei zusätzlichen 40-Pin-Anschlüsse zu benutzen, die auf den Anschlüssen A, B und C montiert sind. Dadurch wird der Verschleiß auf leicht ersetzbare und billige Bauelemente verschoben.

LED-Monitoren, Schalter und Impulsgeber

Acht LED-Monitoren, acht Logikschalter und zwei Impulsgeber dienen der Experimentier-Platine als bequeme Eingabe (Schalter und Impulsgeber) und Anzeigevorrichtung (LED-Monitoren). Alle Logikschalter und Impulsgeber sind mit je zwei NOR-Gatter entprellt. Die entprellte Schaltereinheit mit den entsprechenden Impulsgebern zeigen die Bilder 2-9 und 2-10. Die Schalter SW0 . . . SW7 führen am Ausgang entweder den Zustand logisch 0 oder logisch 1. Die Ausgänge der entprellten Schalter sind mit je einem Federkontakt im Anschluß A verbunden. Die Kontakte sind mit SWn bezeichnet, wobei n gleich 0, 1 . . . 7 ist. Die Impulsgeber (Bild 2-20) geben nur einen einzigen Impuls an die mit Pn und $\overline{Pn}$ bezeichneten Ausgänge; dabei ist n Ausgang 0 oder Ausgang 1. Der Impuls von Pn ist negativ. Beide Impulsgeber werden durch Betätigen ihrer Schalter aktiviert; die Schalter kehren durch Federkraft wieder in ihre Ruhestellung zurück. Die LED-Monitoren sind ausgeschaltet (LEDs bleiben dunkel), wenn die Eingänge mit logisch 0 verbunden sind. Sie schalten ein (LEDs leuchten), wenn die Eingänge an logisch 1 liegen. Bild 2-11 zeigt die komplette LED-Monitor-Schaltung.
Die Tabelle 2-2 gibt die Belastung für den Ein- und Ausgang der entsprechenden Schaltung an.
Fan-In (Eingangslastfaktor): Der Ausdruck "Fan-In" (auch Eingangsfächer genannt) kennzeichnet die Belastung, welche der Eingang eines ICs auf den Ausgang des vorhergehenden ICs ausübt. Zur Vereinfachung der Berechnung ist eine allgemeingültige Zahlenskala maßgebend. So bedeutet bei der TTL-Familie z.B. 1 U.L. (Unit Load = Einheitsladung) einen zulässigen Eingangsstrom von 1,6 mA bei logisch 0 und 0,4 mA bei logisch 1.
Fan-Out (Ausgangslastfaktor): Der Ausdruck "Fan-Out" (auch Ausgangsfächer genannt) kennzeichnet die Leistung, welche am Ausgang eines ICs zur Verfügung steht. Auch hierbei gilt die bereits erwähnte Zahlenskala. 10 U.L. bedeuten am Ausgang im logischen Zustand 0 einen Strom von

Tabelle 2-2. Fan-In und Fan-Out für Logikschalter, Impulsgeber und LED-Monitoren.

Bauelemente auf der Experimentier-Platine		Fan-In		Fan-Out	
		high (log. 1)	low (log. 0)	high (log. 1)	low (log. 0)
Logikschalter				10 U.L.	5 U.L.
Impulsgeber	Pn			10 U.L.	5 U.L.
	$\overline{Pn}$			85 U.L.	15 U.L.
LED-Monitor		0,5 U.L.	0,25 U.L.		

16 mA und bei logisch 1 insgesamt 0,4 mA. Anders ausgedrückt: Der Ausgang mit einem Fan-Out von 10 U.L. kann 10 IC-Eingänge mit einem Fan-In von 1 U.L., oder 5 Eingänge mit einem Fan-In von 2 U.L., usw. steuern.

+5V-Strom-Indikator

Bei dem +5V-Strom-Indikator handelt es sich lediglich um zwei Bauelemente: eine Leuchtdiode (LED) und einen Widerstand. Die LED leuchtet auf, wenn an die Experimentier-Platine eine Spannung von +5 V gelangt. Die Schaltung ist in Bild 2-12 dargestellt.
Ein häufiger Fehler ist die fehlende Versorgungsspannung. In diesem Fall ist die Anzeige-LED dunkel. Verdrahtungsfehler sind ebenfalls für nichtfunktionierende Schaltungen häufig die Ursache. Dabei können u.a. Kurzschlüsse in der Versorgungsspannung auftreten. Die Anzeige-LED bleibt in diesem Fall ebenfalls dunkel. Gegen den Kurzschluß selbst ist das System geschützt und sofort funktionsfähig, wenn der Verdrahtungsfahler beseitigt ist.

FÜR VERSUCHE ERFORDERLICHE BAUELEMENTE

In diesem Kapitel finden Sie in Tabelle 2-3 eine Aufstellung aller Bauelemente einschließlich Drähte, die Sie für sämtliche in diesem Buch beschriebenen Versuche benötigen.

DIE TTL-FAMILIE

Wie aus der Liste der Bauelemente in Tabelle 2 hervorgeht, handelt es sich bei den 7400-TTL-ICs, welche für die in diesem Buch beschriebenen Versuche benötigt werden, um die *Low-Power-Schottky (LS)-Unterfamilie der TTL-Serie (TTL = Transistor-Transistor-Logic).*
Die meisten Hersteller von TTL-ICs halten sich an das Numeriersystem 7400, d.h., alle TTL-ICs haben Teil-Nummern in nachstehender Form:

74XXNNN

wobei XX eine Unterfamilie mit keinem, einem oder zwei Buchstaben und NNN eine zwei- oder dreistellige Ziffer ist. Beispiele für solche TTL-Bezeichnungen gibt die Tabelle 2-3. Die LS-Unterfamilie der TTL-ICs ist bereits erwähnt; es gibt noch mehrere andere. Tabelle 2-4 gibt einen Überblick über die Bezeichnungen, Abkürzungen und Grundeigenschaften.

Alle ICs mit derselben NNN-Zahl sind in der logischen Funktion gleich. So handelt es sich z.B. bei folgenden IC-Bezeichnungen um Vierfach-NAND-Gatter mit zwei Eingängen und gleicher Pinbelegung: 7400, 74H00, 74S00 und 74LS00. Es gibt allerdings auch Fälle, wo die Pinbelegung nicht gleich ist!

Sie stellen sich mit Recht die Frage, ob man für ein in diesem Buch genanntes 74LS-IC auch ein 74-, 74L-, 74H- oder 74S-IC verwenden kann. Die Frage ist nicht so leicht zu beantworten. Sie müssen sich zur Beantwortung die Ein- und Ausgangs-Lastfaktoren Fan-In und Fan-Out sowie die Pinbelegung der betreffenden ICs ansehen.

Beispiele:

1. Bei einem 7400-IC beträgt der maximale Eingangsstrom I_{IL} im Low-Zustand 1,6 mA und im High-Zustand I_{IH} 0,04 mA. Für diese Werte beträgt der Eingangslastfaktor – das Fan-In – insgesamt 1.
2. Für das 74LS00 gilt: $I_{IL} = 0{,}36$ mA und $I_{IH} = 0{,}02$ mA. Folglich hat das 74LS00 einen Low-Eingangslastfaktor von

$$\frac{0{,}36 \text{ mA}}{1{,}6 \text{ mA}}$$

oder 0,225 U.L. und einen High-Eingangslastfaktor von

$$\frac{0{,}02 \text{ mA}}{0{,}04 \text{ mA}}$$

oder 0,5 U.L.

3. Der Ausgangsstrom des 7400 beträgt im Zustand logisch 0 maximal 16 mA und 0,8 mA bei logisch 1. Folglich ist der Ausgangslastfaktor bei log. 0

$$\frac{16 \text{ mA}}{1{,}6 \text{ mA}}$$

oder 10 U.L. und der Ausgangslastfaktor bei log. 1 ist

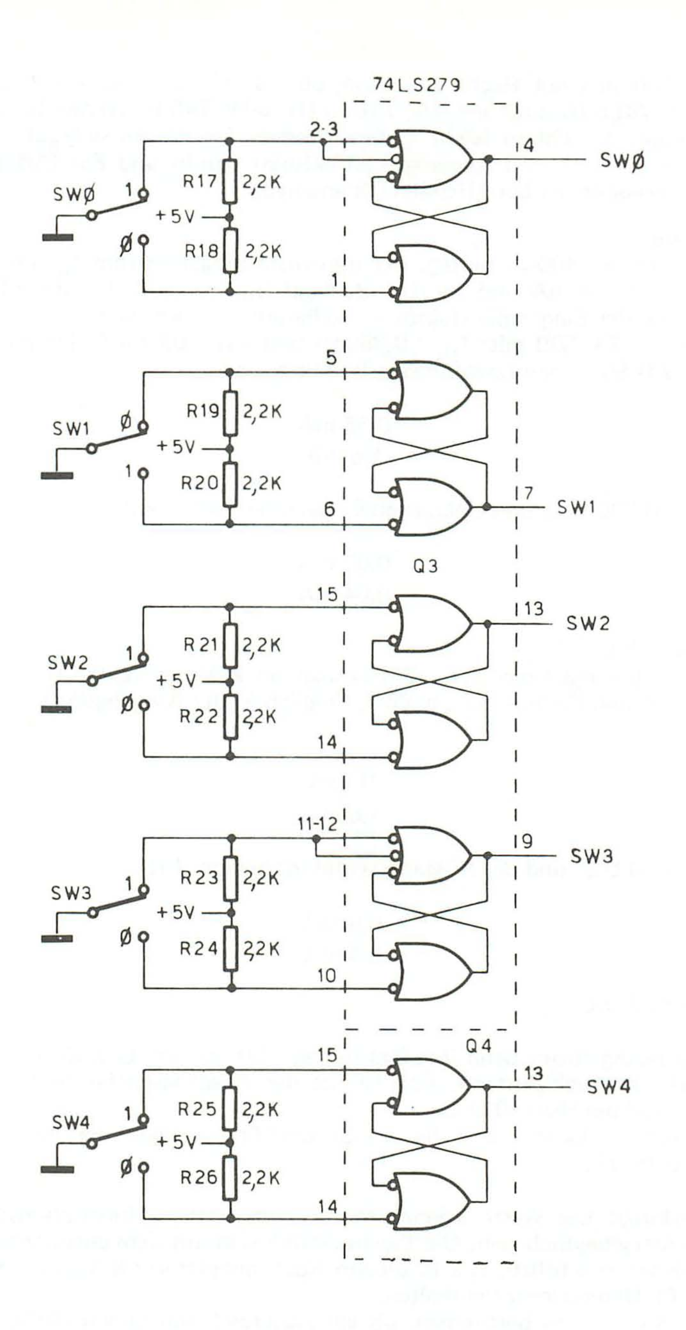
$$\frac{0{,}8 \text{ mA}}{0{,}4 \text{ mA}}$$

oder 20 U.L.

Der Ausgangsstrom beim 74LS00 beträgt 8,0 mA im Low-Zustand und 0,4 mA im High-Zustand. Folglich ist der Ausgangslastfaktor bei Low 5 U.L. und bei High 10 U.L.
Die relativen Lastfaktoren für die Grund-TTL-Familien sind aus Tabelle 2-5 ersichtlich.

Anmerkung: Die Werte können für ICs mit mittlerer Integrationsdichte sehr unterschiedlich sein. Die Eigenschaften sind aus dem entsprechenden Datenblatt ersichtlich. Die in diesem Buch benutzten LS-ICs sind alle in den TTL-Datenbüchern enthalten.
Der beste Weg zu bestimmen, ob ein Austausch mit einem Nicht-LS-IC möglich ist, sind die nachstehenden Prüfungen sowohl im Zustand logisch 0 als auch im Zustand logisch 1 (low und high):

1. ist der Eingangslastfaktor entsprechend hoch?


74LS279
2-3
4
SWØ
R17
2,2K
SWØ
1
+5V
Ø
R18
2,2K
1
5
R19
2,2K
SW1
Ø
+5V
1
R20
2,2K
6
7
SW1
Q3
15
13
SW2
R21
2,2K
SW2
1
+5V
Ø
R22
2,2K
14
11-12
9
SW3
R23
2,2K
SW3
1
+5V
Ø
R24
2,2K
10
Q4
15
13
SW4
R25
2,2K
SW4
1
+5V
Ø
R26
2,2K
14

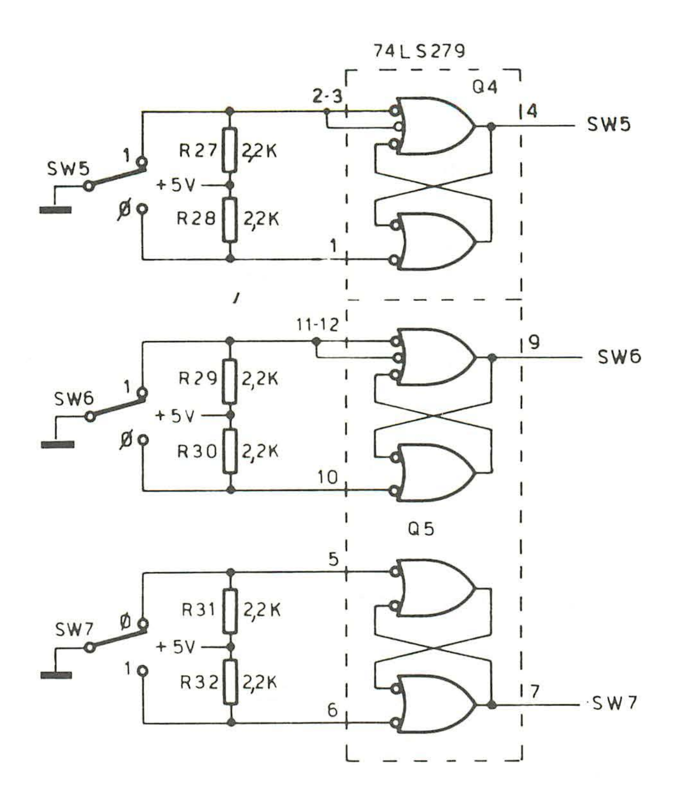


Bild 2-9. Logikschalter SW0 . . . SW7.

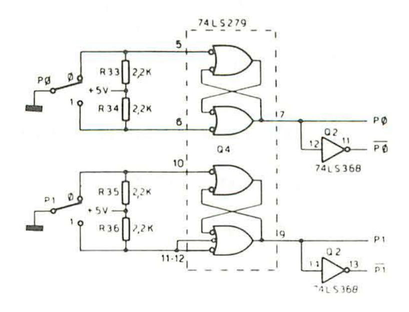


Bild 2-10. Impulsgeber Pn und $\overline{\text{Pn}}$.

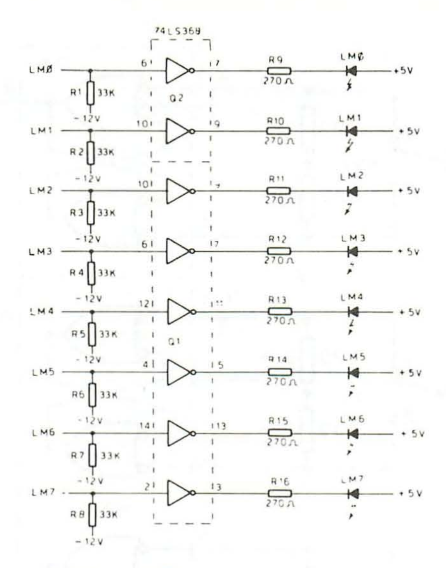


Bild 2-11. LED-Monitoren LM0 . . . LM7.

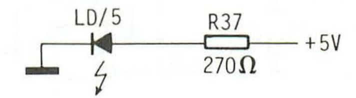


Bild 2-12. Betriebsanzeige für +5V.

2. Liefert der Ausgang für die folgenden ICs genügend Treiberstrom, stimmt der Lastfaktor?

Besteht das für den Austausch vorgesehene Bauelement diese beiden Prüfungen, kann man es ohne Befürchtungen einsetzen.
Die einfachste Alternative ist natürlich die, alle erforderlichen Bauteile vorher zur Hand zu haben. Die LS- sind etwas schneller als gewöhnliche TTL- und verbrauchen weniger Strom.

Tabelle 2-3. Draht- und Bauelementliste

Drahtliste (K1Z80)					
Farbe	**25 cm**	**15 cm**	**10 cm**	**5 cm**	**3 cm**
Grün	4	12	8	8	
Gelb	4	12	8	8	
Schwarz			8	8	10
Rot			8	8	10
Blau	2				

Bauelementliste (K2Z80)

Menge	Gerät	Beschreibung
1	74LS02	Vierfaches NOR-Gatter mit je zwei Eingängen
1	74LS04	Sechsfacher Inverter
1	74LS05	Sechsfacher Inverter (offener Kollektorausgang)
1	74LS08	Vierfaches AND-Gatter mit je zwei Eingängen
1	74LS30	NAND-Gatter mit 8 Eingängen
1	74LS32	Vierfaches OR-Gatter mit je zwei Eingängen
1	74LS42	BCD-zu-Dezimal-Dekoder
1	74LS74	Zweifaches Speicher-Flipflop (D-Flipflop)
1	74LS90	Dezimalzähler
1	74LS125	Vierfaches AND-Leistungsgatter (Low-Freigabe)
1	74LS139	Zweifacher Dekoder/Demultiplexer 1-zu-4
2	74LS175	4-Bit Flipflop getaktet
2	74LS365	Sechsfache Pufferstufe mit gemeinsamer Freigabe
1	Z80-PIO	Parallel-I/O-Interface-IC Z80
1	Z80-CTC	Zähler-Taktgeber-Schaltungs-IC Z80
1	555	astabiler/monostabiler Multivibrator
2	2101A	statisches RAM-IC 4 X 256
3	LED, rot	Leuchtdiode
3	330 Ohm, 1/4 Watt	Widerstände
6	1k, 1/4 Watt	Widerstände
1	33k, 1/4 Watt	Widerstände
1	1000 pF	Kondensator

EINFÜHRUNG IN DIE VERSUCHE

Um Sie mit den Logikschaltern, LED-Monitoren und Impulsgebern auf der Nanocomputer®-Experimentier-Platine vertraut zu machen, sind zwei Versuche vorgesehen. Ein Teil des ersten Versuchs besteht darin zu zeigen, wie der Strom ordnungsgemäß zur Lochrasterplatte SK-10 gelangt.
Es wird empfohlen, die Doppelanschlüsse zum Verdrahten der an A, B und C zu leitenden Signale zu benutzen. Die Drahtstärke von 1 mm hat sich für die Rasterplatte SK-10 als optimal erwiesen. Ein etwas stärkerer Draht (z.B. 1,5 mm) ist noch annehmbar, alle Stärken darüber sind jedoch nicht empfehlenwert, dadurch werden die Anschlüsse A, B und C viel schneller verschlissen.

Tabelle 2-4. Überblick über die TTL-Unterfamilien

Name der Unter-familie	Abk. der Unter-familie	Gatter-Laufzeit	Leistung pro Gatter	Max. Takt-freq.	Bemerkung
Regular-TTL	–	10 ns	10 mW	35 MHz	meist verwendetes IC, sehr niedriger Preis
High-Power-TTL	H	6 ns	22 mW	50 MHz	wird durch neuere Schottky-TTL ersetzt, die schneller ist und weniger Strom verbraucht
Low-Power-TTL	L	33 ns	1 mW	3 MHz	wird ersetzt durch die CMOS-Logik-Familien
Schottky-TTL	S	3 ns	19 mW	125 MHz	bestes Geschwindigkeits/Stromverhältnis
Low-Power Schottky	LS	10 ns	2 mW	45 MHz	besseres Geschw./Stromverhält. als normale TTL-ICs.

Tabelle 2-5.

Familie	Eingangslast		Ausgangslast	
	High	Low	High	Low
74	1,0 U.L.	1,0 U.L.	20 U.L.	10,0 U.L.
74H	1,25 U.L.	1,25 U.L.	25 U.L.	12,5 U.L.
74S	1,25 U.L.	1,25 U.L.	25 U.L.	12,5 U.L.
74LS	0,5 U.L.	0,25 U.L.	10 U.L.	5,0 U.L.

VERSUCH NR. 1

Sinn dieses Versuches ist es, den Einsatz der Logikschalter, LED-Monitoren und Impulsgeber an der Nanocomputer®-Platine zu demonstrieren. Darüber hinaus wird beschrieben, wie man die Lochrasterplatte SK-10 mit Strom versorgt.

Schaltschema

SW0 ———————— LM0
SW1 ———————— LM1
SW2 ———————— LM2
SW3 ———————— LM3
PO ———————— LM6
PO ———————— LM7

1. Schritt
Der 1. Schritt besteht in jedem Versuch darin (auch wenn nicht ausdrücklich erwähnt) festzustellen, ob die Lochrasterplatte SK-10 richtig mit Strom versorgt werden kann. In Bild 2-13 sind alle Drahtanschlüsse aufgeführt, die zur Stromversorgung des SK-10-Platine erforderlich sind. Drei Schaltdrahtpaare verbinden die lötfreien Anschlüsse an den Rändern der Lochrasterplatte zu einer Masse- und Versorgungsleitung.

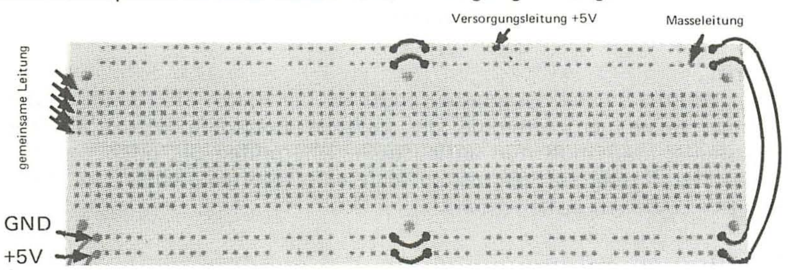


Bild 2-13. Stromversorgung der Lochrasterplatte SK-10.

Das Potential der beiden Leitungen wird durch den Wert an den Strom-Anschlußpins der Verbindungseinheit A bestimmt. Die innere Leitung ist mit einem an GND (Masse) bezeichneten Pin angeschlossen, während die äußere Leitung an den mit +5V bezeichneten Anschluß A, Pin 39, gekoppelt ist. Benutzen Sie für die in Bild 2-13 gezeigten Anschlüsse schwarzen und roten Draht (schwarz = Masse, rot = +5V). Anstatt des schwarzen Drahtes ist auch ein gelb-grün gestreifter Draht zulässig.

2. Schritt
Verdrahten Sie die Schaltung wie am Anfang des Versuchs gezeigt. Verbinden Sie SW0 mit LM0, SW1, mit LM1 usw. Beachten Sie, das hierfür nur die Anschlußeinheit A in Frage kommt; keine Drähte an die Rasterplatte SK-10 anschließen! Für die Signalanschlüsse werden grüne (SW0) und gelbe (SW1) Drähte empfohlen.

3. Schritt
Alle Logikschalter in Stellung OFF (Aus) oder logisch 0 bringen und die Stromversorgung anschließen und einschalten. Prüfen Sie, ob die Monitor-LEDs leuchten.
Der Impulsgeber PO befindet sich in der Regel im Zustand logisch 0, so daß nur die LED LM7 leuchtet.

4. Schritt
Schalter SW0, SW1, SW2 und SW3 auf verschiedene Logikpegel stellen und die korrekte Funktion anhand der LED-Monitoren LM0, LM1, LM2 und LM3 prüfen!

5. Schritt
Impulsgeber PO mehrmals hintereinander betätigen; was betätigen Sie?
Sie beobachten, daß zwar immer eine Diode leuchtet, aber LM6 und LM7

nie gleichzeitig. LM7 leuchtet auf, wenn der Pulsgeber-Schalter in Ruhe- bzw. Nullstellung steht, während LM6 aufleuchtet, wenn man den Pulsgeber-Schalter in die obere bzw. "1"-Stellung bringt.

6. Schritt
Ist die Lochratwerplatte SK10 mittlerweile an die Stromversorgung angeschlossen? Die Antwort muß lauten: Ja. Die entsprechenden Verbindungen haben Sie bereits im 1. Schritt hergestellt.

7. Schritt
Prüfen Sie nun die Versorgungsleitungen an der Platte SK-10. Verbinden Sie einen beliebigen Anschluß der äußeren Lochreihe mit LM5. Die Leuchtdiode LM5 muß nun aufleuchten. Überprüfen Sie noch verschiedene andere Anschlüsse der beiden äußeren Versorgungsleitungen. Dabei muß LED LM5 stets aufleuchten!

8. Schritt
Prüfen Sie die innere Versorgungsleitung (Masse), indem Sie LM5 über einen Draht mit beliebigen Anschlüssen der Masseleitung verbinden; dabei darf LED LM5 in keinem Fall aufleuchten.

9. Schritt
Wie kan man nun mit Hilfe einer einzigen LED des Monitors LM0 . . . LM7 feststellen, ob sich ein Schaltungspunkt im Zustand logisch 0 oder logisch 1 befindet?

Dazu braucht man lediglich eine LED des Monitors mit dem zu prüfenden Schaltungspunkt zu verbinden. Leuchtet die entsprechende LED auf, ist der Schaltungspunkt logisch 1; bleibt die LED dunkel, ist der Schaltungspunkt im Zustand logisch 0.

VERSUCH NR. 2

Zweck dieses Versuches ist zu demonstrieren, wie die Versuchs-Software in Form von zwei EPROMs 2708 geladen wird. Damit man die über 2000 Bytes Software dieses Versuchs nicht manuell eintasten muß, sind sie in zwei EPROMs 2708 gespeichert. Diese EPROMs finden zusammen mit dem 2k-Betriebssystem des Nanocomputers® auf der NBZ80-Platine Platz.
Verfügen Sie nicht über die in den EPROMs gespeicherte Versuchs-Software, muß das Programm manuell geladen werden. Die Tabelle A-1 im Anhang A enthält die absolute Speicheradresse für das entsprechende Programm. Anhang B zeigt die gesamte Versuchs-Software. Aus dem Listing ist das für jeden Versuch relevante Programm ersichtlich.

1. Schritt
Führen Sie die beiden EPROMs 2708 in die vorgesehenen Fassungen neben dem Betriebssystem auf der Nanocomputer®-Platine ein, wobei die Stromversorgung abgeschaltet sein muß! Prüfen Sie die richtige Lage der beiden EPROMs, indem Sie

1. sich davon überzeugen, daß Pin 1 des Versuchs-Software-Eproms so ausgerichtet ist wie Pin 1 des Betriebssystem-EPROMs;
2. feststellen, ob keine Anschlußpins verbogen, sondern alle korrekt in die Fassung eingesteckt sind.

2. Schritt
Strom zum Nanocomputer® einschalten. F000 ins PC-Register einspeichern und G0 drücken. Was beobachten Sie?

Auf dem 7-Segment-Display erscheint in Laufschrift folgender Satz:

SGS-ATES NANO ROUTINES RELEASE 2-2 LOADED CIAO

Diese Anzeige bleibt solange bestehen, bis man die RESET-Taste drückt.

3. Schritt
Durch die Betätigung der RESET-Taste kehrt der Nanocomputer® zu seinen Betriebssystem zurück. Das Programm für die Laufschrift beginnt bei NANOR2. Die vollständige Auflistung finden Sie im Anhang.
Im Kapitel 5 folgt eine ausführliche Bescheibung der Tastatur/Anzeigeeinheit I/O. Sie werden dann in der Lage sein, Ihre eigenen Nachrichten zu schreiben.

4. Schritt
Prüfen Sie verschiedene Speicherstellen, beginnend bei 0100. Stellen Sie fest, ob alle Bytes eingespeichert sind:

Stelle	Inhalt
0100	D3
0101	C5
0102	18
0103	FC
0104	3E
0105	21

5. Schritt
Der Programmstart bei der Adresse F000 hat einen Block-Transfer zur Folge. Der EPROM-Inhalt F000 . . . F7FF wird dabei in das RAM ab 0100 eingeschrieben. Wie Sie bei den restlichen Versuchen in diesem Buch feststellen, ist die Software mit nur geringfügigen Änderungen in einigen Fällen gebrauchsfertig eingegeben.

Nachstehend ein Speicherplan des EPROM:

JP zum Block-Bewegungsprogram
NANOCOMPUTER EXPERIMENT SOFTWARE
Block-Bewegungsprogram
Nachrichtentreiber vollständig eingeben
Nachrichten-Daten
unbenutzte EPROM-Bytes

Kapitel 3

Synchronisations-Impulserzeugung: Adressen- und Geräte-Auswahlimpulse

Dieses Kapitel behandelt eine der vier Hauptaufgaben des Z-80-Interfacing, nämlich die Erzeugung von Synchronisations-Impulsen für die programmierte Ein- und Ausgabe von Daten zur und von der CPU. Weitere Synchronisations-Impulse, wie die Bestätigung von Interrupts und BUS-Anforderungen, werden in späteren Kapiteln erörtert. Außerdem lernen Sie einige Dekodierungs-Schaltungen kennen, die zur Erzeugung der Geräte- und Adressen-Auswahlimpulse erforderlich sind.

Am Ende dieses Kapitels sind Sie in der Lage:

- Geräte- und Adressen-Auswahlimpulse zu definieren;
- die Begriffe *Hardware* und *Software* zu definieren;
- zu erklären, was "Austausch von Software gegen Hardware" bedeutet;
- die Z-80-Befehle zu erörtern, die für die Erzeugung der Geräte- und Adressen-Auswahlimpulse verantwortlich sind;
- die Z-80-Synchronisations-Impulse $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ und $\overline{\text{WR}}$ sowie einige davon abgeleitete Signale, $\overline{\text{IN}}$, $\overline{\text{OUT}}$, $\overline{\text{MEMR}}$ und $\overline{\text{MEMW}}$ zu definieren und anzuwenden;
- Schaltkreise schematisch darzustellen, die zur Erzeugung der Adressen- und Geräte-Auswahlimpulse benutzt werden können;
- mehrere verschiedene Anwendungsarten für Geräte-Auswahlimpulse aufzuführen.

HARDWARE UND SOFTWARE BEIM Z-80-INTERFACING

In diesem und den folgenden Kapiteln werden Sie wiederholt auf die Begriffe Hardware und Software stoßen; sie sollen daher gleich zu Anfang definiert werden:

Hardware ist ein Sammelbegriff für die mechanischen, magnetischen, elektronischen und elektrischen Bauelemente eines Computers; dazu gehört auch der mechanische Aufbau all dieser Bauelemente.

Software ist ein Sammelbegriff für die Gesamtheit von Programmen und Routinen, die zum Ausbau der Leistungsfähigkeit eines Computers erforderlich sind z.B. Compiler, Routinen und Subroutinen; das Gegenteil von Hardware.

Den Computer mit all seinen Bauelementen Drähten, Platinenhilfsmittel und externen Geräten bezeichnet man mit Hardware. Die von Ihnen geschriebenen Programme und Subroutinen sind die Software. Das Betriebssystem des Computers gehört ebenfalls zur Software.

Das zweite Kapitel definiert das Interfacing des Z-80 als ein System, das die Übertragung von digitalen Daten zwischen der CPU und den externen Geräten synchronisiert. Diese Synchronisation wird fast immer durch eine Koordination von Software mit Hardware erzielt. Es ist durchaus nicht ungewöhnlich, wenn ein Z-80-Programmierer immer wieder die Schaltbilder des Computers zu Rate zieht, für die er die Software schreibt. Um brauchbare Ergebnisse zu erzielen, müssen die (durch das Schaltbild verkörperte) Hardware und die (durch das Programm ververkörperte) Software eng miteinander verknüpft sein.

SPEICHER-ZUGRIFFSBEFEHLE UND DAS $\overline{\text{MREQ}}$-SIGNAL

Wie bereits erwähnt, liest der Nanocomputer® die Objektcodes der gespeicherten Programme der Reihe nach ab, um sie anschließend zu entschlüsseln. In einem Programm bedeutet die Ausführung eines Befehls das Interfacing mit einem externen Gerät, nämlich dem Speicher. Das Zeitdiagramm des M1- oder Objektcode-Abrufzyklus (Bild 3-1) zeigt, wie die Z-80-CPU der "Außenwelt" mitteilt, daß sie mit dem Ablesen des nächsten Programms beginnen möchte:

1. Das $\overline{\text{M1}}$-Signal wird aktiviert, logisch 0.
2. Die Speicheradresse gelangt auf den Adreß-BUS.
3. Das $\overline{\text{MREQ}}$-Signal ist ebenfalls aktiviert, sobald es auf logisch 0 geht. Dies ist allerdings erst dann der Fall, NACHDEM sich die Speicheradresse auf dem Adreß-BUS voll stabilisiert hat.
4. Schließlich wird noch das $\overline{\text{RD}}$-Signal aktiv (logisch 0).

Die "Außenwelt", besonders das Speichersystem, muß diese Nachricht empfangen, übersetzen und das Notwendige veranlassen. Die acht Daten-Bits müssen auf den Daten-BUS, damit die richtige Information zur Verfügung steht, wenn die CPU den Daten-BUS abliest.
Bevor die genaue Beschreibung der Lese-Signale folgt, ist das Wissen um die Befehlsausführung der CPU erforderlich. Betrachtet wird z.B. der Befehl LD A, (HL) mit der Speicheradresse 0100 im HL-Register. Der Hex-Code für LD A, (HL) ist 7E. Der erste Objektcode-Abruf bzw. M1-Zyklus liest das Byte 7E aus dem Speicher in die CPU ein. Nach der Entschlüsselung des Befehls weiß die CPU, daß sie

1. einen Ein-Byte-Befehl ausführen muß;
2. den Inhalt der durch das Registerpaar HL adressierten Speicherstelle in den Akkumulator laden muß.

Zur Information für die "Außenwelt" stellt die CPU folgende Signale bereit:

1. Die Adresse des Speicherplatzes steht auf dem Adreß-BUS zur Verfügung.
2. Das $\overline{\text{MREQ}}$-Signal geht auf logisch 0, wird also aktiviert.
3. Ebenfalls aktiviert (log. 0) wird das $\overline{\text{RD}}$-Signal.

Mit Ausnahme der Aktivierung von $\overline{M1}$ ist der Objektcode-Abruf identisch mit dem Speicher-Lesezyklus, den der LD A,(HL)-Befehl auslöst. Da die Hauptaufgabe des M1-Signals darin besteht, die Objektcode-Entschlüsselung mit der Byte-Ablesung zu koordinieren, erfolgt die Kommunikation zwischen Speicher und CPU in der Regel über $\overline{MREQ}$, $\overline{RD}$ (für Speicher-Ablesungen), $\overline{WR}$ (für Speicherübertragungen) sowie den Daten- und Adreß-BUS. Merken Sie sich die folgenden beiden Punkte:

1. Speicherzugriffe kommen sowohl durch die *Ausführung* von Befehlen als auch durch die *Anforderung* von Befehlen zustande.
2. Die Signale zum *Anfordern* von Objektcodes und zur *Ausführung* von Speicher-Lesebefehlen sind im wesentlichen identisch.

Im nächsten Beispiel liest die CPU den Befehl LD (HL), A mit dem entsprechenden Hex-Code 77. Ist zur Ausführung dieses Befehls die Kommunikation mit einem externen Gerät erforderlich? Wenn ja, welches Gerät wird angesprochen und welche Signale stellt die CPU an den entsprechen Anschlußpins zur Verfügung? Der LD (HL), A-Befehl veranlaßt die CPU den Akkumulator-Inhalt in die Speicherstelle einzulesen, die der Inhalt des HL-Registerpaares anweist. Enthält das HL 0100, muß der Akkumulator-Inhalt in die Speicherstelle 0100 umgeschrieben werden. Diese Stelle befindet sich *außerhalb* der CPU. Es ist also eine Kommunikation mit der "Außenwelt" erforderlich, wobei die CPU folgende Signale auf die entsprechenden Anschlußpins legt:

1. 0100 gelangt auf den Adreß-BUS und zeigt an, in welche Speicherstelle die Daten umgeschrieben werden.
2. Das $\overline{MREQ}$-Signal wird aktiviert, da das externe Gerät der Speicher ist.
3. Der Akkumulator-Inhalt muß zur Umschreibung in die Speicherstelle 0100 auf dem Daten-BUS zur Verfügung stehen.
4. Das $\overline{WR}$-Signal ist ebenfalls aktiviert, weil WR (WRITE – Schreiben) die gewünschte Operation ist.

Wie Sie sehen, besteht zwischen dem Speicher-*Lese-* und dem Speicher-*Einschreib*-Vorgang ein Unterschied. Während sich beim Ablesen das Signal $\overline{RD}$ aktiviert, schaltet sich beim Einschreiben das $\overline{WR}$-Signal ein. Ein feinerer Unterschied besteht in der zeitlichen Folge der Datenauflage auf den Daten-BUS. Bei einer Ablese-Operation legt das externe Gerät Daten auf den Daten-BUS zum Ablesen durch die CPU. Bei der Einschreib-Operation legt die CPU Daten auf den Daten-BUS zum Ablesen durch das externe Gerät. Die Zeitunterschiede für diese beiden Operationen sind in den Zeitdiagrammen der Bilder 3-1 und 3-2 leicht zu erkennen.
Wie in Bild 3-3 dargestellt, kann man beide Zeitdiagramme überlagern. Diese Überlagerung ist die übliche Technik zur Erleichterung von Zeitvergleichen.
Die feinen Zeitunterschiede zwischen Speicher-Ablese und Speicher-Einschreib-Operationen sind bereits erwähnt. Besonders kritisch ist die relative Zeitfolge des Auflegens von Daten auf den Daten-BUS und die Aktivierung der synchronisierten Impulse $\overline{MREQ}$ und $\overline{RD}$. Bei der Ablese-Operation sind $\overline{MREQ}$ und $\overline{RD}$ bereits aktiviert, bevor der Speicher die Daten auf den Daten-BUS legt! Im Gegensatz dazu ist bei einer Schreib-Operation $\overline{WR}$ nach Auflage der Daten auf den Daten-BUS aktiv. Bei einer Speicher-Schreib-Operation ist es kritisch, die richtigen Daten und Adres-

sen auf den Daten-BUS zu bekommen, *bevor* der Speicher einen SCHREIB-Befehl erhält.

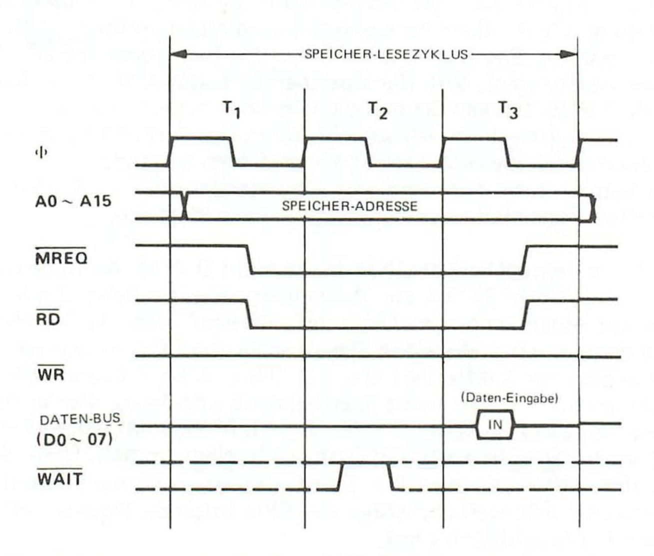


Bild 3-1. Z-80-Zeitdiagramm eines Speicher-Lesezyklus ohne Wartezustände.

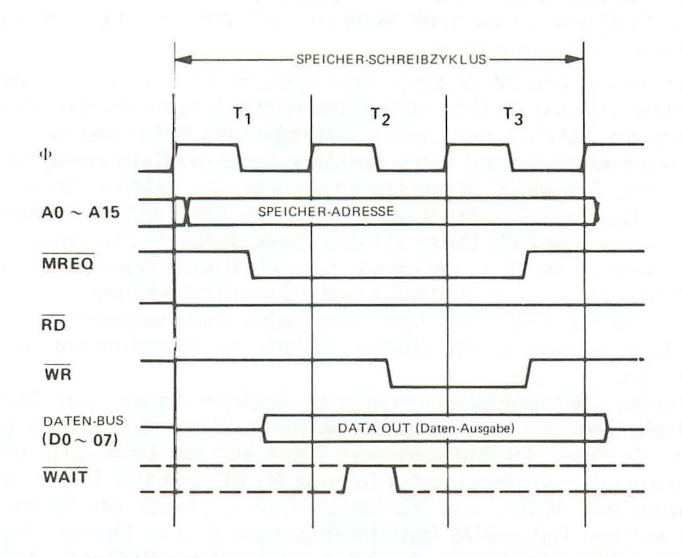


Bild 3-2. Z-80-Zeitdiagramm eines Speicher-Schreibzyklus ohne Wartezustände.

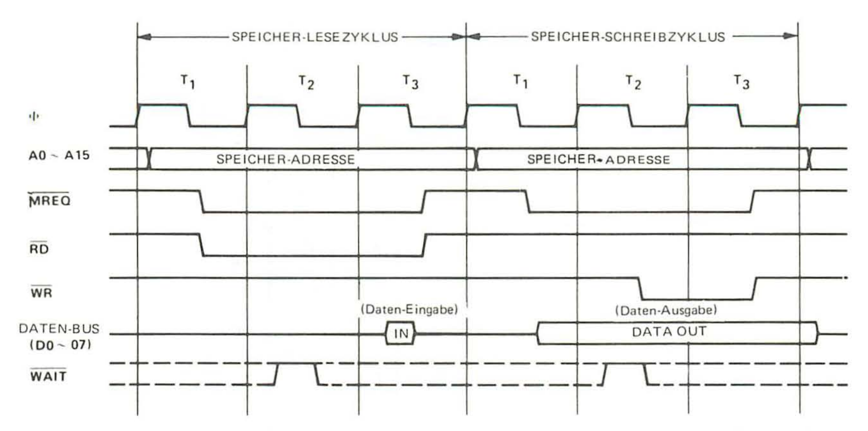


Bild 3-3. Z-80-CPU-Zeitdiagramm eines Speicher-Lese- bzw. Speicher-Schreibzyklus ohne Wartezustände.

Da die Daten- und Adreß-Leitungen in der Regel gepuffert und die Adreß-Leitungen entschlüsselt sind, unterliegen beide einer gewissen Zeitverzögerung. Damit die Speicher-ICs richtig arbeiten, ist es also erforderlich, das $\overline{WR}$-Signal zu verzögern. Es darf erst dann am "R/W"-Pin des dynamischen Speichers eintreffen, *nachdem* die Daten und Adressen sprechenden BUS-Leitungen vollständig zur Verfügung stehen.
Für solche "kitzligen" Zeitprobleme erzeugt der Nanocomputer® das $\overline{DBWR}$-Signal. Es handelt sich dabei um ein ca. 100 ns verzögertes $\overline{BWR}$-Signal. Es ist allgemein sicherer, für alle Versuche mit der Experimentier-Platine $\overline{DBWR}$ zu verwenden.
Probieren Sie beide Signale selbst aus, um festzustellen, ob es irgendwelche Leistungsunterschiede gibt. Die Schaltung zur Erzeugung des $\overline{DBWR}$ aus dem $\overline{BWR}$-Signal ist in Bild 2-8 dargestellt.
Bevor Sie sich dem folgenden Abschnitt zu wenden, noch eine kurze, rückblickende Frage: Muß die CPU bei dem Befehl LD A,B mit einem externen Gerät – dem Speicher – korrespondieren? Stehen während der Befehlausführung an den Anschlußpins der CPU irgendwelche Synchronisationsimpulse zur Verfügung?
Die Antwort auf diese Frage lautet: Nein. Der Befehl LD A,B bedeutet "Lade den Akkumulator mit dem Inhalt des Registers B". Beide Register sind Bestandteil der CPU, so daß für die Umschreibung des Inhalts von Register A in Register B keine Hilfe von außen notwendig ist.

I/O-BEFEHLE UND DAS $\overline{IORQ}$-SIGNAL

Außer dem Speicher gibt es viele externe Geräte, die über das Interfacing mit der Z-80-CPU verbunden sind. Die Tastatur des Nanocomputers® ist ein Eingabegerät, während die 7-Segment- und LED-Anzeigen zum Ausgabegerät gehören, das mit der Z-80-CPU über das Interfacing verbunden ist. Wird eine Z-80-CPU in eine Geschirrspülmachine eingebaut, dann ist die Geschirrspülmaschine ein Gerät, das über das Interfacing mit der CPU gekoppelt ist. Weitere Geräte, die z.B. mit den Z-80-CPUs verknüpft sind:

				SOURCE PORT ADDRESS	
				IMMED.	REG. INDIR.
				(n)	(C)
INPUT DESTINATION	INPUT 'IN'	REG ADDRESSING	A	DB n	ED 78
			B		ED 40
			C		ED 48
			D		ED 50
			E		ED 58
			H		ED 60
			L		ED 68
	'INI' – INPUT & Inc HL, Dec B	REG. INDIR	(HL)		ED A2
	'INIR' – INP, Inc HL, Dec B, REPEAT IF B≠0				ED B2
	'IND' – INPUT & Dec HL, Dec B				ED AA
	'INDR' – INPUT, Dec HL, Dec B, REPEAT IF B≠0				ED BA

BLOCK INPUT COMMANDS ('INI', 'INIR', 'IND', 'INDR')

Bild 3-4. Eingabe-Befehlsgruppe.

Fernschreiber, andere Z-80-CPUs, graphische Sichtgeräte, Lochstreifenlese- und Dekodiergeräte, Drucker, Plattenlaufwerke, Kassetten-Rekorder und Tonbandgeräte, Musiklautsprecher, Laborinstrumente sowie eine Vielzahl von anderen Geräten. Zu jedem externen Gerät gehört eine spezielle Software, mit deren Hilfe die Z-80-CPU die einzelnen Geräte (einschl. Speicher) unterscheidet. Den einwandfreien Betrieb koordinieren die erforderlichen Synchronisationsimpulse. Die Befehlsgruppen, die Synchronisationsimpulse für externe Geräte – mit Ausnahme des Speichers – erzeugen, heißen EINGABEGRUPPE (Bild 3-4) und AUSGABEGRUPPE (Bild 3-6). Das folgende Kapitel befaßt sich mit diesen beiden Gruppen.

EINGABE-BEFEHLSGRUPPE

Die Z-80-CPU hat im Gegensatz zur 8080A nicht nur einen Eingabebefehl (Objektcode DB), sondern mehrere. Dies gibt der Z-80-CPU erweiterte Anwendungsmöglichkeiten. Bei der folgenden Beschreibung der einzelnen Z-80-Eingabebefehle werden nachstehende Abkürzungen verwendet: r steht für eins der Register A, B, C, D, E, H oder L; n gibt eine zweistellige Hex-Ziffer (8 Bits pro Wort) an. Die folgenden Drei-Bit-Codes werden mit dem Objektcode vereint, um das Bestimmungsregister anzuzeigen:

r	Register
000	B
001	C
010	D
011	E
100	H
101	L
110	unbenutzt (Flag = Merkbit setzen)
111	A

IN A, (n)

Dieser Befehl besagt: Lade Daten in Form von 8-Bit-Worten von dem externen Gerät mit dem Code n in den Akkumulator.
Der Hex-Code für diesen Befehl ist DB n, wobei n der zweistellige Hex-Code ist. Zur Ausführung dieses Befehls verrichtet die Z-80-CPU folgende Operationen:

1. Auf die 8 niederwertigen Bits des Adreß-BUS A0 . . . A7 stellt die CPU den Gerätecode n abrufbereit.
   Auf die 8 höherwertigen Bits des Adreß-BUS A8 . . . A15 stellt die CPU den Akkumulator-Inhalt abrufbereit.
2. Aktivierung der Signale $\overline{IORQ}$ und $\overline{RD}$.
3. Einlesen der von Gerät n als Antwort auf die obigen Signale auf den Daten-BUS gelegten Daten in den Akkumulator.

Das Zeitdiagramm für den Eingabezyklus ist in Bild 3-5 dargestellt.
Der IN A,(n)-Befehl ist der einzige vom 8080A-Mikroprozessor ausgeführte Befehl. Anstatt den Akkumulator-Inhalt auf die obere Hälfte des Adreß-BUS zu legen, überträgt der 8080A den Gerätecode auf A8 . . . A15. Dieser Unterschied beeinflußt die Hardware des 8080A gegenüber dem Z-80-Interfacing, hat aber in der Regel keinen Einfluß auf die Kompatibilität der Software für die beiden CPUs. Es sind keine Schaltbilder für 8080A- oder Z-80-Eingabeschaltungen bekannt, für die der Unterschied zu Inkompatibilitäten führen würde, obgleich eine Inkompatibilität nicht auszuschließen ist.
Alle Z-80-Eingabe- und Ausgabebefehle benutzen die höherwertigen acht Bits des Adreß-Bus, ohne sie lediglich zu überschreiben. Dies ist ein Versuch, dem externen Schaltkreis möglichst viele Informationen zur Verfügung zu stellen. In diesem Fall bildet der Akkumulatorzustand vor Ausführung des IN A,(n)-Befehls auf der oberen Hälfte des Adreß-BUS eine Information, die für die Interface-Schaltung von Nutzen sein kann. Mögliche

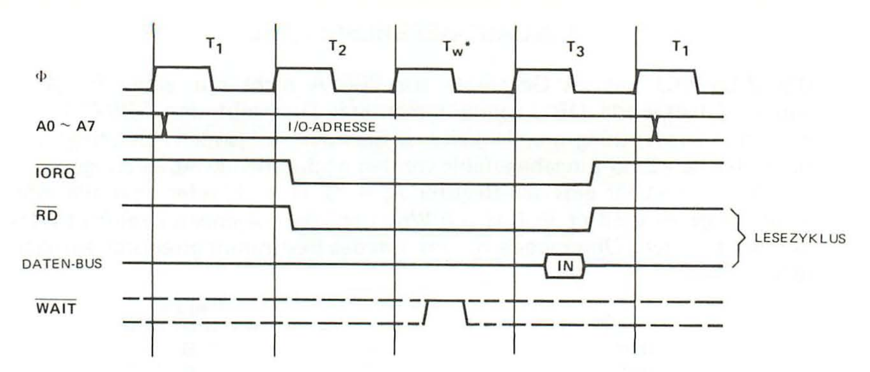


Bild 3-5. Zeitfolge eines Ein-/Ausgabe-Lesezyklus mit der automatisch eingefügten Wartezeit T_W*.

Nutzanwendungen dieser Informationen sind:

- abwechselndes Ein- und Auslesen von Bytes mit dem Eingabegerät, falls das Gerät solche Operationen durchführen kann;
- Abtasten anderer Schaltungen, um Operationen auszuführen, die mit der Eingabe-Operation gemeinsam auszuführen sind wie z.B. das Rücksetzen von Zustands-Flags, Anhalten laufender Operationen, Auslösen verwandter Operationen, Abtasten von Zähler- oder Zeittaktschaltungen usw.

IN r,(C)

Dieser Befehl macht die Daten-Eingabe-Operationen noch flexibler. Er ermöglicht, die indirekte Registeradressierung für die Spezifikation der Gerätecodes einzusetzen. Es kann das gewünschte Eingabegerät adressiert werden, indem man den Gerätecode in das C-Register eingibt. Der Gerätcode ist kein "festverschlüsseltes" Byte des Befehls, da er in dem oben erörterten IN A,(n)-Befehl enthalten ist. Außerdem läßt sich mit diesem Befehl das Bestimmungsregister für die Eingabedaten festlegen. Beispiel:

```
ED 40        IN B,(C)
```

überträgt die Daten aus dem Gerät, die der 8-Bit-Code des Registers C adressiert, in das Register B.

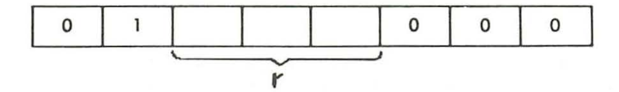
Die Z-80-CPU veranlaßt für die Ausführung dieses Befehls folgende Operationen:

1. Auflage des Inhalts von Register C auf die niedrigstwertigen 8 Bits des Adreß-BUS A0 . . . A7; Auflage des gegenwärtigen Inhalts von Register B auf die höherwertigen acht Bits des Adreß-BUS A8 . . . A15;
2. Aktivierung der Signale $\overline{IORQ}$ und $\overline{RD}$.
3. Einlesen der Daten, die vom externen Gerät durch Adressierung des Gerätcodes in Register C auf dem Daten-BUS bereitstehen, ins Register.

Die Zeitfolge dieser Operationen geht ebenfalls aus Bild 3-5 hervor.

Der Objektcode für den IN r,(C)-Befehl enthält zwei Bytes. Das erste Byte ist stets ED. Das zweite Byte enthält auf den Bitstellen D3 . . . D5 den

Registercode, der dem Bestimmungsregister r entspricht. Das zweite Byte entspricht also dem folgende Schema:



Der 3-Bit-Code 110 ist keinem der Register A, B, C, D, E, H und L zugeordnet. Der resultierende Befehl ED 70 beeinflußt lediglich die Flags und keine anderen Register.

Während der Akkumulator-Eingabebefehl IN A,(n) keine Flags beeinflußt, setzt der IN r,(C)-Befehl die Null-, Paritäts- und Vorzeichen-Flags nach den Ergebnissen der Operation. Die Additions/Subtraktions- (N) und Half-Carry-Flags (H) werden beide zurückgesetzt.
Jeder Ein-/Ausgabebefehl schiebt automatisch einen mit T_W* bezeichneten Wartezustand ein, um einem externen Gerät eine zusätzliche Antwortzeit einzuräumen. Weitere Einzelheiten über das WAIT-Signal folgen in dem Kapitel ''WARTEZUSTÄNDE''.

BLOCK-EINGABEBEFEHLE: INI, INIR, IND, und INDR

Die Z-80 unterstützt die Block-Eingabebefehle ähnlich wie die in Buch 1 beschriebenen Block-Transfer- und Block-Suchbefehle. Vor der Ausführung eines Block-Eingabebefehls muß ein Z-80-Programm die Register B, C und HL wie folgt starten:

B = Zahl der einzugebenden Bytes
C = Gerätecode des Eingabegeräts
HL = Startadresse für die Folge von Eingabe-Bytes

INI

Für die Durchführung des INI-Befehls (input-increment) = Eingabe-Erhöhung sind folgende Schritte erforderlich:

1. Ein Byte des durch Register C adressierten Geräts wird in die Speicherstelle eingegeben, die der Inhalt des Registerpaares HL anweist. Das heißt auf die CPU und die Interface-Schaltung bezogen:
   a) Der Gerätecode in Register C gelangt auf die niederwertigen 8 Bits des Adreß-BUS (A0 . . . A7). Der Inhalt von Register B – der Byte-Zähler – auf die höherwerigen 8 Bits des Adreß-BUS (A8 . . . A15).
   b) Die Signale $\overline{IORQ}$ und $\overline{RD}$ werden aktiviert.
   c) Die o.g. Signale lesen in die CPU die von dem externen Gerät auf den Daten-BUS gelegten Daten ein. Das Eingabe-Byte wird in der vom HL angewiesenen Speicherstelle gespeichert (die Abspeicherung des Eingabe-Bytes in (HL) löst einen Speicher-Schreibzyklus aus).
2. Der Inhalt des Registers B wird um 1 vermindert (dekrementiert).
3. Der Inhalt des Registerpaares HL verringert sich ebenfalls um 1.

4. Ist der B-Registerinhalt nach der Dekrementierung 0, wird das Null-Flag gesetzt, anderenfalls zurückgesetzt. In jedem Fall gesetzt wird das N-Flag (Additions/Subtraktions-Flag).

INIR

Zur Ausführung des INIR-Befehls (input-increment-repeat = Eingabe-Erhöhung-Wiederholung) sind folgende Schritte erforderlich:

1. Ein Byte des vom Inhalt des Registers C adressierten Geräts wird in die vom Inhalt des Registerpaares HL angewiesene Speicherstelle eingegeben:

   $$(HL) \leftarrow (C)$$

   Die CPU und die Interface-Schaltung arbeiten in der gleichen Weise wie bei INI unter Punkt c beschrieben.
2. Das B-Register dekrementiert um 1.
3. Der Inhalt des Registerpaares HL wird um 1 erhöht (inkrementiert).
4. Ist der Inhalt von Register B nicht gleich 0, wiederholen sich die Schritte 1, 2 und 3. Das Null-Flag wird zurückgesetzt. Ist B gleich 0, geht das Null-Flag auf logisch 1 (wird gesetzt), der Nanocomputer® setzt das Programm mit dem nächsten Befehl fort. In allen Fällen nimmt das N-Flag den Zustand logisch 1 an.

IND und INDR

Die Ausführung der Befehle IND (input-decrement = Eingabe-Verminderung) und INDR (input-decrement-repeat = Eingabe-Verminderung-Wiederholung) erfolgt in ähnlichen Schritten. Der einzige Unterschied ist die Verminderung (Dekrementierung) von HL im 3. Schritt. Somitgeben INI und INIR Byte-Folgen in den Speicher nach obenein (die Richtung der Speicheradressen ist positiv), während die IND- und INDR-Befehle die Byte-Folgen in den Speicher nach unten eingeben (die Richtung der Speicheradressen ist negativ).
Die Nützlichkeit der BLOCK-EINGABEBEFEHLE is wohl jedem Anwender deutlich; deshalb wird an dieser Stelle auf weitere Beispiele verzichtet.
Das folgende Kapitel befaßt sich mit der AUSGABEBEFEHLSGRUPPE. Es beschreibt die Hardware des Interfacing und insbesondere die dazu erforderlichen Schaltungen. Sie dekodieren die von der CPU kommenden Signale bei der Ausführung von Ein- und Ausgabe-Befehlen (IN- und OUT-Befehlen). Softwarebeispiele zu den Ein- und Ausgabegruppen beschreibt das Buch an anderer Stelle.

AUSGABE-BEFEHLSGRUPPE

Wie aus Bild 3-6 ersichtlich, setzt sich die AUSGABEGRUPPE aus Befehlen zusammen, die analog zu den EINGABEGRUPPE-Befehlen sind. Nachstehend eine Aufstellung der Ausgabebefehle mit den analogen Eingabebefehlen:

SOURCE

			REGISTER							REG. IND.
			A	B	C	D	E	H	L	(HL)
'OUT'	IMMED.	(n)	■							
'OUT'	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI' – OUTPUT Inc HL, Dec b	REG. IND.	(C)								ED A3
'OTIR' – OUTPUT, Inc HL, Dec B, REPEAT IF B≠0	REG. IND.	(C)								ED B3
'OUTD' – OUTPUT Dec HL & B	REG. IND.	(C)								ED AB
'OTDR' – OUTPUT, Dec HL & B, REPEAT IF B≠0	REG. IND.	(C)								ED BB

BLOCK OUTPUT COMMANDS (OUTI, OTIR, OUTD, OTDR)

PORT DESTINATION ADDRESS (IMMED./REG. IND. columns)

Bild 3-6. Ausgabe-Befehlsgruppe.

Ausgabebefehl	Eingabebefehl
OUT (n),A	IN A,(n)
OUT (C),r	IN r,(C)
OUTI	INI
OTIR	INIR
OUTD	IND
OTDR	INDR

Es bestehen drei grundsätzliche Unterschiede zwischen Befehlen der Eingabegruppe und der Ausgabegruppe:

1. Die Richtung des Datenflusses in bezug auf die CPU.
2. Ein Ausgabebefehl aktiviert das $\overline{WR}$-Signal, während der analoge Eingabebefehl das $\overline{RD}$-Signal aktiviert.
3. Auf Grund der unterschiedlichen Rolle der CPU als Empfänger (Eingabe) oder Sender (Ausgabe) gibt es Zeitunterschiede.
   In Bild 3-7 sind die Zeitfolgen für Ein- und Ausgabezyklen in einem Diagram dargestellt. Wie Sie sehen, benötigt die CPU für die Auflage der Ausgabedaten auf den Daten-BUS eine verhältnismäßig lange Zeit, verglichen mit der Zeit, die ein externes Gerät dafür braucht. Darum hat die CPU bei einer Eingabe-Operation einen viel engeren Fehlerspielraum als ein externes Gerät während einer Ausgabe-Operation der CPU. Die $\overline{IORQ}$-Signal-Zeitfolge ist für beide gleich.
4. Weil in Mikrocomputersystemen strenge Zeitanforderungen in bezug auf die relativen Ankunftszeiten der Daten- und Synchronisationsimpulse bei Ein-/Ausgabe-/Schreibzyklen notwendig sind, kann ein verzögertes $\overline{WR}$-Signal (bei der Nanocomputer®-Experimentier-Platine $\overline{DBWR}$ erforderlich sein. Der Wunsch nach einem $\overline{DBWR}$-Signal für Speicher-Schreibzyklen besteht ebenso bei Ein-/Ausgabe-/Schreibzyklen.

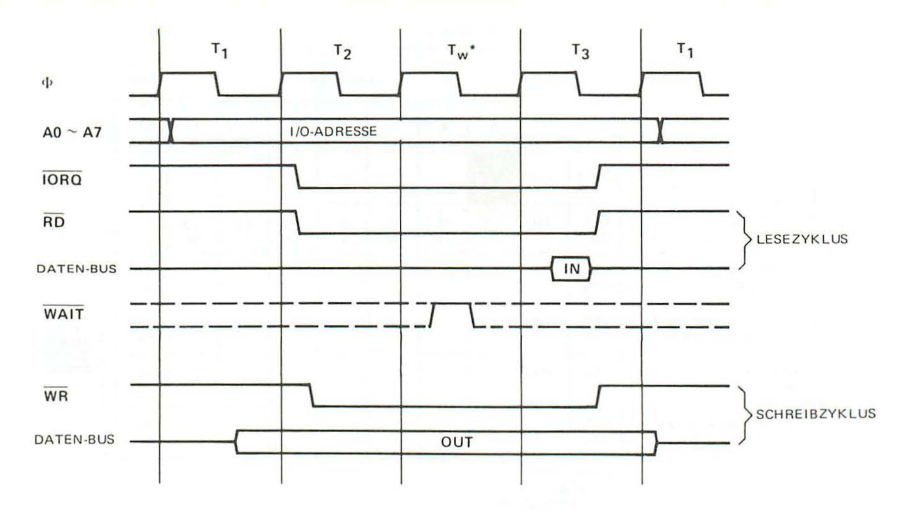


Bild 3-7. Zeitfolge der Z-80-CPU für Ein- und Ausgabezyklen.

Die Überlagerung der Ein- und Ausgabesignale in Bild 3-7 dient lediglich der besseren Übersicht. Die Ein-/Ausgabe-/Lese- und Ein-/Ausgabe-/Schreib/-Zyklen können in Wirklichkeit nicht hintereinander erfolgen. Die Beschreibung der automatisch eingefügten Wartezustände T_W* folgt im Abschnitt ''WARTEZUSTÄNDE''.

Eine Kurzbeschreibung aller Ein- und Ausgabebefehle ist in Tabelle 3-1 wiedergegeben.

Die Tabelle zeigt in den Spalten von links nach rechts die mnemonische Assemblersprache, die allgemein symbolische Kurzschreibweise, den Inhalt der Flag-Register nach der Befehlsausführung, den binären Op-Code, die Byte-Nummer, die Anzahl der Speicherzyklen, die Gesamtanzahl der Taktzyklen bei der Befehlsausführung. In der achten Spalte sind die Befehle – falls erforderlich – mit Bemerkungen versehen.
Zum besseren Verständnis sind die in der Tabelle verwendeten Begriffe nachfolgend ins deutsche übersetzt:
if – wenn; if r = 110 only the flags will be affected – wenn r = 110 werden nur die Flags verändert; repeat until – Wiederholung bis . . . Die unter der Tabelle stehenden Erklärungen lauten:
Wenn das Ergebnis von B-1 Null ist, Flag setzen; ansonsten zurücksetzen.

Flag-Bezeichnung: • = Flag nicht betroffen, 0 = Flag zurücksetzen, 1 = Flag setzen, X = Flag nicht beeinflußt, ↕ Flag ist je nach Operationsergebnis betroffen.

$\overline{\text{IN}}$-, $\overline{\text{OUT}}$-, $\overline{\text{MEMR}}$- und $\overline{\text{MEMW}}$-Signale

Die Bedeutung der Signale $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{MREQ}}$ und $\overline{\text{IORQ}}$ bei der Synchroni-

Tabelle 3-1. Ein- und Ausgabe-Gruppenbefehle

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	•	•	•	•	•	11 011 011 ← n →	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	•	↕	P	↕	0	↕	11 101 101 01 r 000	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	•	↕ (1)	X	X	1	X	11 101 101 10 100 010	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	•	1	X	X	1	X	11 101 101 10 110 010	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	•	↕ (1)	X	X	1	X	11 101 101 10 101 010	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B - 1 HL ← HL -1 Repeat until B = 0	•	1	X	X	1	X	11 101 101 10 111 010	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	•	•	•	•	11 010 011 ← n →	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	•	•	•	•	11 101 101 01 r 001	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	•	↕ (1)	X	X	1	X	11 101 101 10 100 011	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	•	1	X	X	1	X	11 101 101 10 110 011	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	•	↕ (1)	X	X	1	X	11 101 101 10 101 011	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B - 1 HL ← HL -1 Repeat until B = 0	•	1	X	X	1	X	11 101 101 10 111 011	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Notes: (1) If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

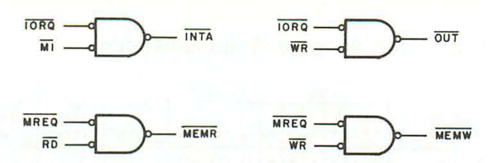


Bild 3-8. Logisches Symbol und boolesche Algebra für die Signale $\overline{MEMR}$, $\overline{MEMW}$, $\overline{INTA}$ und $\overline{OUT}$.

sation Z-80-programmierter Ein-/Ausgaben kann man wie folgt zusammen fassen:

– $\overline{MREQ}$ und $\overline{RD}$ aktiv	MEMory Read operation (Speicher-Leseoperation)
– $\overline{MREQ}$ und $\overline{WR}$ aktiv	MEMory Write operation (Speicher-Schreiboperation)
– $\overline{IORQ}$ und $\overline{RD}$ aktiv	I/O INput operation (I/O-Eingabeoperation)
– $\overline{IORQ}$ und $\overline{WR}$ aktiv	I/O OUTput operation (I/O-Ausgabeoperation)

Bei allen Mikrocomputer-I/O-Schaltungen ist es nützlich, die aufgeführten Synchronisationsimpulse zu benutzen. Sie sind in Bild 3-8 näher definiert Die aus der Wahrheitstabelle 3-2 ersichtliche Verknüpfung für das MEMR-Signal gilt auch für alle anderen in Bild 3-8 aufgeführten Signale.
So ist z.B. nach der folgenden Wahrheitstabelle $\overline{MEMR}$ nur dann aktiv (logisch 0), wenn sowohl $\overline{MREQ}$ als auch $\overline{RD}$ aktiv (logisch 0) sind.

Tabelle 3-2. Wahrheitstabelle für die Erzeugung des $\overline{MEMR}$-Signals

$\overline{MREQ}$	$\overline{RD}$	$\overline{MEMR}$
0	0	0 (aktiv)
0	1	1
1	0	1
1	1	1

Nachstehend eine Zusammenfassung der Hauptpunkte aus den letzten Abschnitten:

– Die Z-80 liest externe Speicher aus und nimmt folglich mit einem anderen Gerät durch den Objektcode-Abruf oder M1-Zyklus Verbindung auf, um alle Befehle eines Programmes aus dem Speicher auszuführen.

– Man unterscheidet zwei Hauptgruppen von Z-80-Befehlen, zu deren Ausführung die CPU mit externen Geräten Verbindung aufnehmen muß (Interfacing):

a) Die $\overline{MREQ}$-Gruppe: Befehle, die Zugriff zum Speicher haben um Lese- oder Schreiboperationen durchzuführen ($\overline{RD}$ oder $\overline{WR}$) wie z.B. LD (HL),A, INC (HL), oder AND (HL).

b) Die $\overline{IORQ}$-Gruppe: $\overline{IN}$- und $\overline{OUT}$-Befehle.

- Alle Befehle, welche die CPU veranlassen mit einem externen Gerät Verbindung aufzunehmen (Interfacing), legen entweder einen Adreß- oder Gerätecode auf den Adreß-BUS, aktivieren zwei der Steuersignale $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ und $\overline{WR}$ und schreiben Daten auf den Daten-BUS oder lesen sie von ihm ab.
- Für Speicher-Schreib- und I/O-Ausgabeoperationen können verzögerte WR-Signale erforderlich sein.
- Vier sehr nützliche unmittelbare Signale in Z-80-Speicherzugriffs- und I/O-Schaltungen sind:

$$\begin{aligned}
\overline{MEMR} &= \overline{MREQ} \cdot \overline{RD} \\
\overline{MEMW} &= \overline{MREQ} \cdot \overline{WR} \\
\overline{IN} &= \overline{IORQ} \cdot \overline{RD} \\
\overline{OUT} &= \overline{IORQ} \cdot \overline{WR}
\end{aligned}$$

ZEITFOLGEBERECHNUNGEN FÜR Z-80 I/O-BEFEHLE

In diesem und den vorigen Kapiteln sind verschiedene Zeitdiagramme abgedruckt. Diese Diagramme zeigen nur die relativen Zeiten für eine Reihe von Ereignissen an, nämlich die Signale auf den drei externen Z-80-BUS-Leitungen. So zeigt z.B. Bild 3-7 das Zeitdiagramm für Ein- und Ausgabeoperationen, bei denen die untere Hälfte des Adreß-BUS A0 . . . A7 die Adresse aufnimmt. Der Adreß-BUS wird mit der ansteigenden Flanke von T_1 aktiviert. Unmittelbar hinter der positiven Flanke von T_2 erfolgt die Aktivierung von $\overline{IORQ}$; die negative Flanke von T_3 bringt das Signal wieder in die Ruhestellung. Bei einer Eingabeoperation stehen die Daten nur relativ kurze Zeit auf dem Daten-BUS zur Verfügung; das ist nur während der Periodendauer T_3 der Fall (Bild 3-7). Um die Frage zu beantworten, wie lange ein $\overline{IORQ}$-Impuls dauert oder wie lange sich die Eingabedaten auf dem Daten-BUS befinden oder weitere *absolute* Zeitfragen, benötigt man die Impulslänge in Sekunden oder Bruchteilen davon. Die Zeit pro T-Zyklus ist der Kehrwert der Taktfrequenz des Nanocomputers®. Wenn der Computer z.B. mit 1 MHz oder 1.000.000 Taktzyklen pro Sekunde arbeitet, beträgt die Zykluszeit 1/1.000.000 Sekunden pro T-Zyklus oder eine Mikrosekunde. Die schnellsten Z-80 arbeiten mit 4 MHz oder 250 ns (0,250 Mikrosekunden) pro T-Zyklus. In der Regel arbeitet ein Nanocomputer® mit ca. 2,5 MHz. Wie lange dauert ein T-Zyklus? Die Antwort lautet 400 ns.

Für den IORQ-Impuls benötigt der Nanocomputer® insgesamt 2,5 T-Zyklen. Das entspricht bei der Arbeitsfrequenz von 2,5 MHz einer Ausführungszeit von 1000 ns oder 1 Mikrosekunde. Die Eingabedaten befinden sich etwas weniger als einen halben T-Zyklus oder 200 ns auf dem Daten-BUS. Das alles sind relativ kurze Zeitabschnitte. Wenn Sie Daten über das Tastenfeld in den Nanocomputer® eingeben, benötigt er nur Bruchteile von Sekunden um sie zu verstehen. Diese präzise Synchronisation ist Grundvoraussetzung aller Digitalcomputer. Die fortschreitende Technik eröffnet die Möglichkeit, die Sekunden in immer kleinere Bruch-

teile zu unterteilen. Dies geht aber nur bis zu einem bestimmten Punkt, denn die Elektronen sind max. so schnell wie das Licht: ca. 30 cm/ns. Die heutige Technik hat den Punkt erreicht, wo die Lichtgeschwindigkeit eine natürliche Grenze darstellt. Diese Grenze macht sich in der Computertechnik negativ bemerkbar. Es entstehen dadurch bei der Signalübermittlung Verzögerungszeiten, die pro 30 cm Draht oder Leitung 1 ns betragen. Dieser Abschnitt hat deutlich gemacht, mit welcher ungeheuren Geschwindigkeit der Nanocomputer® arbeitet und wie wichtig eine präzise Synchronisation der Vorgänge sowohl innerhalb als auch außerhalb der CPU ist.

WARTEZUSTÄNDE

Eine Möglichkeit, die strengen Zeitvorschriften zu lockern, ist die Verwendung von Wartezuständen. Es handelt sich dabei um ereignislose Zeit- bzw. Taktzyklen, die in den Programmablauf eingefügt werden. Während des Wartezustandes bleiben alle Signale unverändert. So beträgt z.B. die Zeit zwischen dem Abtasten einer gegebenen Speicherstelle durch die CPU und dem Ablesen des Daten-BUS bei einem Speicher-Lesezyklus ohne Wartezustand (siehe Bild 3-1) nur ca. zwei T-Zyklen. Das sind 500 ns bei einer Arbeitsfrequenz von 4 MHz. Die ZUGRIFFSZEIT des Speichers muß nicht ebenso schnell sein. Die *Zugriffszeit* ist die vom Speicher benötigte Zeit, mit der CPU in Verbindung zu treten, das adressierte Daten-Byte zu finden und auf den Daten-BUS zu legen. So kann z.B. der Speicher die Daten bei einer seiner Adressen in 1,0 Mikrosekunden auf den Daten-BUS legen. Unter Zuhilfenahme des $\overline{\text{WAIT}}$-Signals (Pin 24) kann der Speicher die CPU dazu auffordern, zwei Extra-T-Zyklen zu warten, bevor sie den Daten-BUS abliest. Selbstverständlich muß die Interfaceschaltung zwischen CPU und Speicher die entsprechende Logik enthalten, um das low-aktive $\overline{\text{WAIT}}$-Signal für den Zeitraum von zwei T-Zyklen auf logisch 0 zu halten. Hat die CPU festgestellt, daß die $\overline{\text{WAIT}}$-Leitung lo-

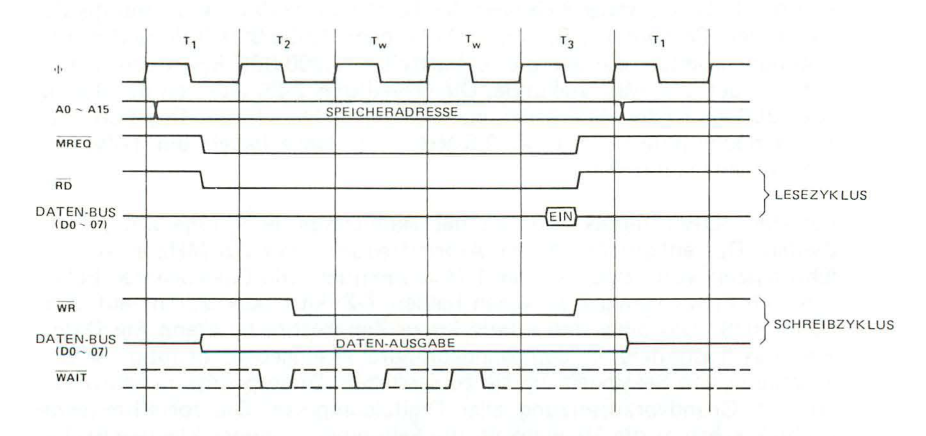


Bild 3-9. Speicher-Lese- oder Schreibzyklen der Z-80-CPU mit Wartezuständen.

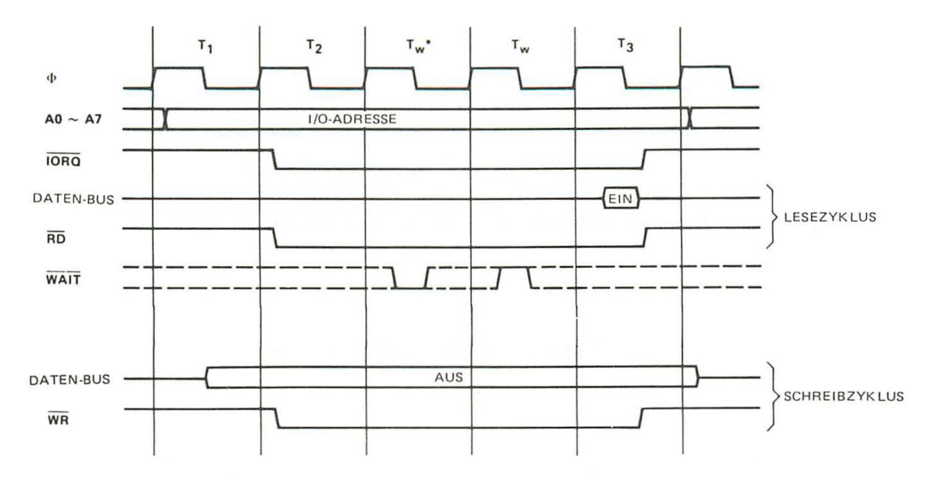


Bild 3-10. Ein- oder Ausgabezyklen der Z-80-CPU mit Wartezuständen.

gisch 0 ist, wartet sie mit dem Ablesen der Daten solange, bis sie wieder hohes Potential (log. 1) führt. Die CPU prüft zuerst die $\overline{\text{WAIT}}$-Leitung an der negativen Flanke von T_2 des Speicher-Lesezyklus und dann je einmal nach jedem folgenden T-Zyklus. Zeitdiagramme für die Speicher- und Geräte-I/O-Zyklen mit entsprechend eingefügten Wartezuständen sind in den Bildern 3-9 und 3-10 dargestellt.

GERÄTE- UND ADRESSEN-AUSWAHLIMPULSE

Zunächst eine Definition der Begriffe *Geräte-Auswahlimpuls und Adressen-Auswahlimpuls:*
Geräte-Auswahlimpuls — ein von der CPU erzeugter Synchronisationsimpuls mit dazugehöriger Dekodierschaltung zur Koordination der Datenübertragung zwischen der CPU und einem externen Gerät außer dem Speicher.
Adressen-Auswahlimpuls — ein von der CPU erzeugter Synchronisationsimpuls mit dazugehöriger Dekodierschaltung zur Koordination der Datenübertragung zwischen der CPU und dem Speicher.
Bei der Z-80 sind die Geräte-Auswahlimpulse mit dem $\overline{\text{IORQ}}$-Signal kombiniert, während die Adressen-Auswahlimpulse mit dem $\overline{\text{MREQ}}$-Signal kombiniert sind. Dieser Abschnitt beschreibt verschiedene Dekodierschaltungen für beide Arten der Auswahlimpulse.

Geräte-Auswahlimpulse

Die Schaltung zur Erzeugung des Geräte-Auswahlimpulses bei der Z-80 verwendet ein Dekoder/Demultiplexer-IC sowie verschiedene Gatter. Diese Schaltung verknüpft das $\overline{\text{IORQ}}$- mit dem $\overline{\text{RD}}$- oder $\overline{\text{WR}}$-Signal, um die niederwertigen acht Bits auf dem Adreß-BUS für das Dekoder/Demultiplexer-IC abzutasten. Das Dekoder/Demultiplexer-IC setzt dann die Geräteadresse ein, um auf einer ihrer Ausgabeleitungen einen Abtastimpuls

Tabelle 3-4. Eigenschaften des T54LS138/T74LS138.

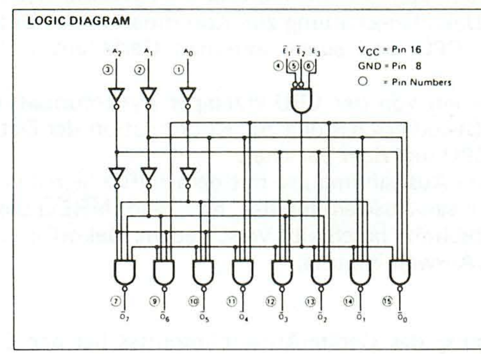
1-OF-8 DECODER/DEMULTIPLEXER

DESCRIPTION – The LSTTL/MSI T54LS138/T74LS138 is a high speed 1-of-8 Decoder/Demultiplexer. This device is ideally suited for high speed bipolar memory chip select address decoding. The multiple input enables allow parallel expansion to a 1-of-24 decoder using just three LS138 devices or to a 1-of-32 decoder using four LS138s and one inverter. The LS138 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- DEMULTIPLEXING CAPABILITY
- MULTIPLE INPUT ENABLE FOR EASY EXPANSION
- TYPICAL POWER DISSIPATION OF 32 mW
- ACTIVE LOW MUTUALLY EXCLUSIVE OUTPUTS
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

PIN NAMES		LOADING (Note a) HIGH	LOW
$A_0 - A_2$	Address Inputs	0.5 U.L.	0.25 U.L.
$\bar{E}_1, \bar{E}_2$	Enable (Active LOW) Inputs	0.5 U.L.	0.25 U.L.
E_3	Enable (Active HIGH) Input	0.5 U.L.	0.25 U.L.
$\bar{O}_0 - \bar{O}_7$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.



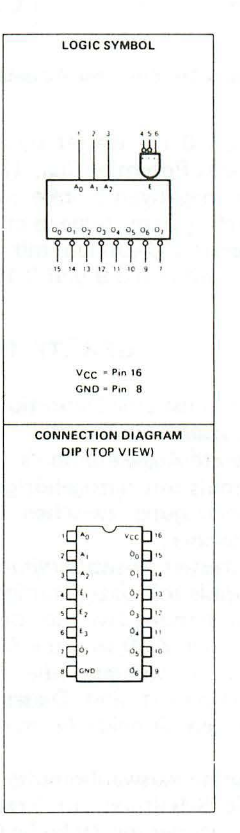


Tabelle 3-4. Fortsetzung

FUNCTIONAL DESCRIPTION – The LS138 is a high speed 1-of-8 Decoder/Demultiplexer fabricated with the low power Schottky barrier diode process. The decoder accepts three binary weighted inputs (A_0, A_1, A_2) and when enabled provides eight mutually exclusive active LOW outputs ($\overline{O}_0$-$\overline{O}_7$). The LS138 features three Enable inputs, two active LOW ($\overline{E}_1$, $\overline{E}_2$) and one active HIGH (E_3). All outputs will be HIGH unless $\overline{E}_1$ and $\overline{E}_2$ are LOW and E_3 is HIGH. This multiple enable function allows easy parallel expansion of the device to a 1-of-32 (5 lines to 32 lines) decoder with just four LS138s and one inverter. (See Figure a.)

The LS138 can be used as an 8-output demultiplexer by using one of the active LOW Enable inputs as the data input and the other Enable inputs as strobes. The Enable inputs which are not used must be permanently tied to their appropriate active HIGH or active LOW state.

TRUTH TABLE

INPUTS						OUTPUTS							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$	$\overline{O}_4$	$\overline{O}_5$	$\overline{O}_6$	$\overline{O}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

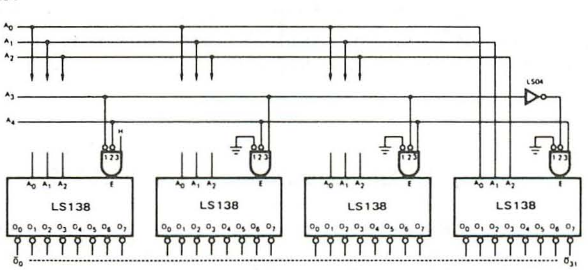


Fig. a.

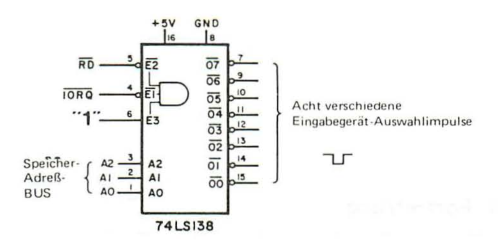


Bild 3-11. Erzeugung von acht verschiedenen Eingabegerät-Auswahlimpulsen mit einem Dekoder/Demultiplexer-IC 74LS138 (mehrdeutige Dekodierung von A0 . . . A15)

zu erzeugen. Zu dem an der abgetasteten Dekoder/Demultiplexer-Ausgabeleitung angeschlossenen Gerät hat die CPU Zugriff. Da die meisten Dekoder/Demultiplexer für weniger als acht Eingaben ausgelegt sind, werden Teilmengen der niederwertigen Adreßleitungen mit jedem Dekoder/Multiplexer benutzt. Nachstehend einige Beispiele:

Beispiel 1: Mehrdeutige Dekodierung

Das IC in Bild 3-11 benutzt die drei niedrigstwertigen Bits des Gerätecodes sowie die $\overline{IORQ}$- und $\overline{RD}$-Signale als Eingabe. Das IC ist nur aktiv, wenn sowohl $\overline{IORQ}$ und $\overline{RD}$ im Zustand logisch 0 sind, also nur bei einer IN-Operation. Die logischen Werte auf den Adreßleitungen A2 bis A0 bestimmen, welcher der acht Ausgänge aktiviert ist (aktiv = logisch 0).

Das Dekoder/Demultiplexer-IC 74LS138 ist ein Chip mit positiver Logik. Die acht Ausgänge sind im nichtaktivierten Zustand logisch 1. Wie bereits erwähnt, legen die logischen Zustände an den Eingängen A0 bis A2 den aktivierten Ausgang fest. Ist z.B. der Eingang A0 logisch 1 und A1 und A2 logisch 0, geht der Ausgang $\overline{01}$ auf logisch 0, wenn ebenfalls die Freigabeimpulse $\overline{IORQ}$ und $\overline{RD}$ logisch 0 sind. Aus der Wahrheitstabelle 3-3 geht der Zusammenhang zwischen den logischen Ein- und Ausgangszuständen deutlich hervor.

Tabelle 3-3. Wahrheitstabelle zum IC 74LS138

Eingänge						aktivierter Ausgang
$\overline{RD}$	$\overline{IORQ}$	"1"	A0	A1	A2	
0	0	1	0	0	0	$\overline{00}$
0	0	1	1	0	0	$\overline{01}$
0	0	1	0	1	0	$\overline{02}$
0	0	1	1	1	0	$\overline{03}$
0	0	1	0	0	1	$\overline{04}$
0	0	1	1	0	1	$\overline{05}$
0	0	1	0	1	1	$\overline{06}$
0	0	1	1	1	1	$\overline{07}$

Die Tabelle 3-4 zeigt eine genauere Beschreibung des ICs. Es ist ein Auszug aus dem Original-Datenblatt und deshalb in englischer Sprache. Das gilt auch für weitere Datenblattauszüge in diesem Buch.
Das IC aus Bild 3-11 kann mit seinen 8 Ausgängen bis zu maximal 8 Eingabegeräte bedienen.

Der negative Ausgangsimpuls veranlaßt also ein Eingabegerät seine Daten in dem Augenblick auf den Daten-BUS zu legen, in dem die CPU den BUS abliest. Die fünf fehlenden Adressen A3 bis A7 des Gerätecodes bleiben bei dieser Schaltungsart unberücksichtigt. Deshalb ist eine Mehrfach-Dekodierung möglich. So aktivieren die Binärcodes folgender Hex-Zahlen immer den Ausgang $\overline{01}$:

01, 08, 11, 18, 21, 28, 31, 38, 41, 48 . . . E1, E8, F1 und F8.

Das bedeutet, das an Ausgang $\overline{01}$ angeschlossene Eingabegerät wird von 32 verschiedenen Gerätecodes adressiert – nicht gerade eine Katastrophe, aber je nach Anwendungsfall unerwünscht. Die gerade beschriebene Dekodierungsart *dekodiert* also den 8-Bit-Gerätecode *mehrdeutig.*

Beispiel 2: Absolute Dekodierung
Die Schaltung in Bild 3-12 dekodiert den 8-Bit-Gerätecode *absolut.* Das heißt: Der Geräte-Auswahlimpuls wird aus den Adreß-Bits A0 bis A7 dekodiert.
Bild 3-12 zeigt, wie jeder der möglichen 256 Eingangsgerät-Auswahlimpulse entsteht. Man benötigt dazu drei ICs 74LS138. Ihnen werden die Freigabe-Impulse $\overline{IORQ}$ und $\overline{RD}$ parallel zugeführt. Die Adreß-Bits A0 . . . A7 teilen sich auf die drei ICs auf. Je ein Ausgang der einzelnen ICs wird nun über ein ODER-Gatter (für negative Auswahlimpulse) oder über ein NOR-Gatter (für positive Auswahlimpulse) miteinander verknüpft. Unter dem Begriff "Geräte-Auswahlimpuls" versteht man also den Impuls am Ausgang eines ODER- oder eines NOR-Gatters, deren Eingänge mit drei Ausgängen der 74LS138 verbunden sind.
Zur Erzeugung des Auswahlimpulses für den Gerätecode A3 (hex) mit einem ODER-Gatter ist folgende Verknüpfung notwendig;

Ausgang 03 (Pin 12) von 74LS138 Nr. 1
(für 011 in A2 . . . A0)
Ausgang 04 (Pin 11) von 74LS138 Nr. 2
(für 100 in A5 . . . A3)
Ausgang 02 (Pin 13) von 74LS138 Nr. 3
(für 10 in A7 . . . A6)

1010 0011 = A3H

Im Folgenden wird ein Geräte-Auswahlimpuls nach seiner Operation mit IN (I/O, Lesen: $\overline{IORQ}$- und $\overline{RD}$-Signale aktiv) oder mit OUT (I/O, Schreiben: $\overline{IORQ}$- und $\overline{WR}$-Signale aktiv) bezeichnet. Für den Gerätecode gilt:

$\overline{IN}$ n oder $\overline{OUT}$ n oder IN n oder OUT n

wobei n der Gerätecode in hexadezimaler Beschreibung ist und der Strich das Signal als low-aktiv ausweist. So wird z.B. der beschriebene Geräte-Auswahlimpuls mit

$\overline{IN\ A3H}$

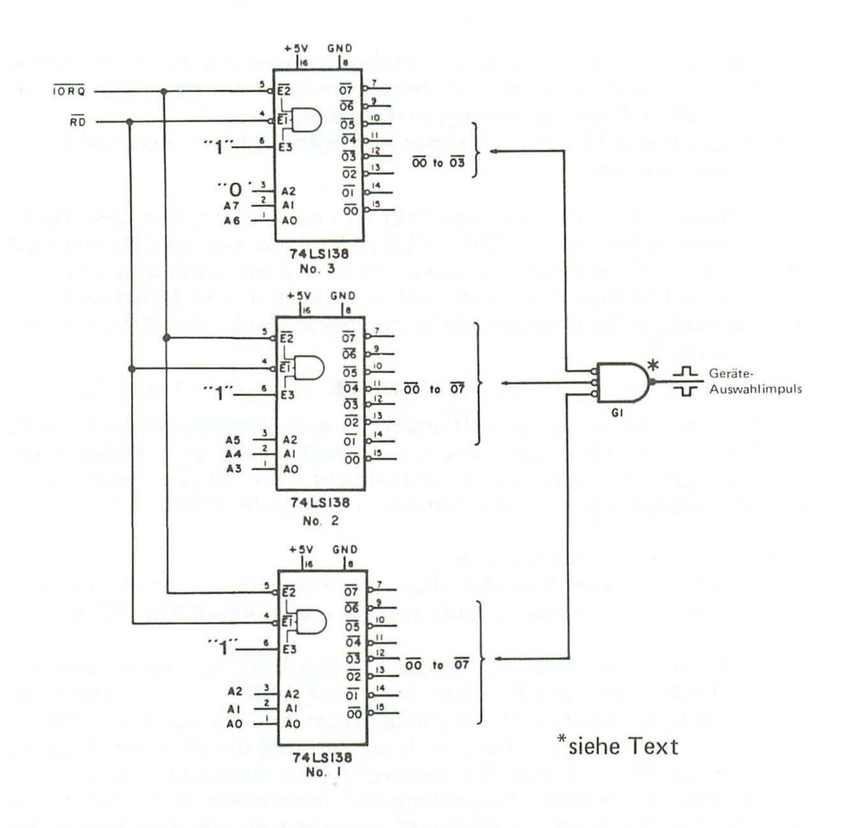


Bild 3-12. Absolute Dekodierungs-Schaltung zur Erzeugung von 256 verschiedenen Geräte-Auswahlimpulsen.

bezeichnet; der Gerätcode ist A3 und der Impuls ist negativ bzw. low-aktiv.

Bild 3-13 zeigt eine Schaltung mit OR- und NOR-Gatter für die Geräte-Auswahlimpulse IN 00H, IN 01H, IN 02H und IN A3H (OR = ODER; NOR = Nicht-ODER).

Bei den meisten Anwendungen ist die genannte Schaltung überdimensioniert, da in der Regel nicht alle 256 möglichen Geräte-Auswahlimpulse erzeugt werden müssen.

Die Geräte-Auswahlimpulse sind definiert als eine Folge von Adressen in einer Teilmenge des gesamten 00- bis FF-Bereichs. Der Nanocomputer® kann eine Teildekodierung des Geräte-Auswahlimpulses für 00 bis 1F (siehe Definition der Leitungen IOQ3, IOEO-3 und IOUO-3 in Kapitel 2 und am Ende dieses Kapitels) erzeugen. Die Schaltungen in Bild 3-11 und 3-12 sind nur dann von Nutzen, wenn die Geräte-Auswahlimpulse in einem System im gesamten Bereich 00 bis FF verstreut sind. Folgen die Gerätecodes aufeinander, sind derartige Schaltungen nicht vorteilhaft.

Beispiel 3: Die Generatorschaltung des Nanocomputers® für den Geräte-Auswahlimpuls

Die Schaltung in Bild 3-14 benutzt ebenfalls das Dekoder/Multiplexer-IC. Die Adressenleitungen BA2, BA3 und BA4 sind mit den Auswahl-Eingängen, die Adressenleitungen BA5, BA6 und BA7 mit den Freigabe-Eingängen verbunden; dem BA6-Eingang ist ein Inverter vorgeschaltet. Die Beschaltung ist anders als bei den bereits bekannten Dekoder-Beispielen, denn die Signale IORQ und WR oder RD spielen bei der Aktivierung des Dekoders in Bild 3-14 keine Rolle. Die $\overline{RD}$- und $\overline{WR}$-Signale werden in Verbindung mit einem Impuls auf den Dekoder/Demultiplexer-Ausgangsleitungen zur Aktivierung des PIO-ICs benutzt.

Diese Schaltung demonstriert eine andere Dekodiertechnik. Sie erzeugt einen Impuls streng nach dem Gerätecode und unterscheidet mit Hilfe der Impulse $\overline{IORQ}$ und $\overline{RD}$ oder $\overline{WR}$, ob es sich um eine IN- oder OUT-Operation handelt. Die Dekodier-Methode hängt von der Zahl der Dekoder/Demultiplexer-ICs ab, die zur Erzeugung der Geräte- bzw. Speicher-Auswahlimpulse erforderlich sind. Kein Dekoder/Demultiplexer widmet

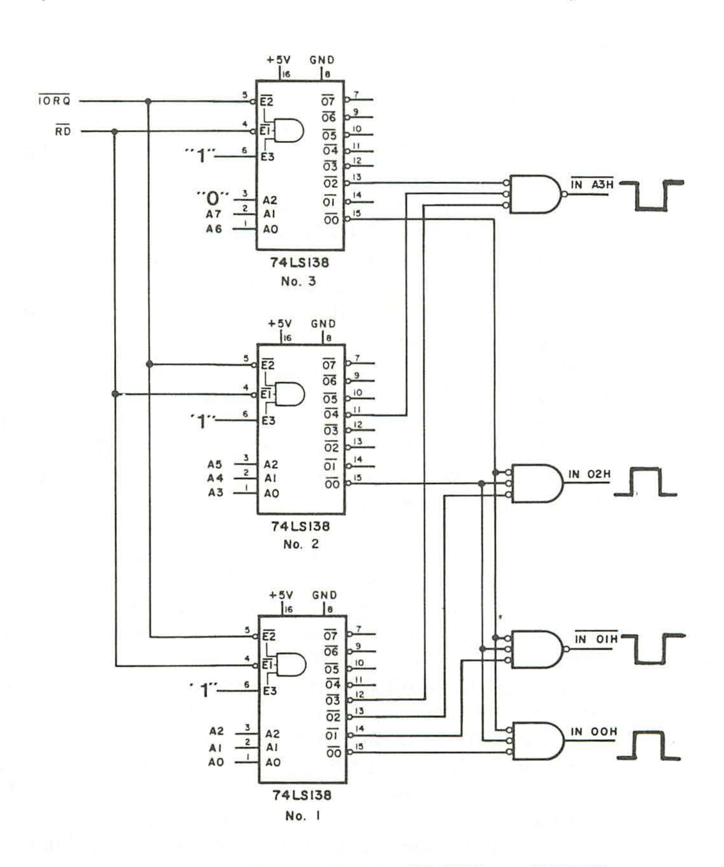


Bild 3-13. Geräte-Auswahlimpulse IN 00H, IN 01H, $\overline{\text{IN 02H}}$ und $\overline{\text{IN A3H}}$.

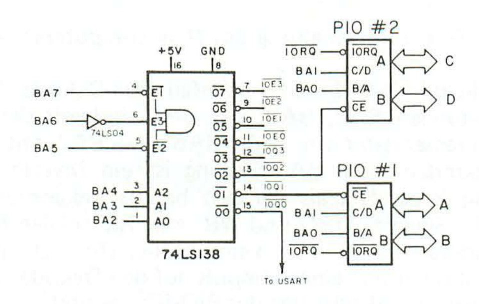


Bild 3-14. Dekodierschaltung für die vom Nanocomputer® benutzten Geräte-Auswahlimpulse.

sich ausschließlich den IORQ-, MEMR-, RD- oder WR-Operationen. In den vorigen Beispielen aktivierten die ICs 74LS138 nur die IN-Operationen!
Da die BA0- und BA1-Adressenleitungen für das PIO-IC Eingänge sind, werden die niedrigerwertigen acht Bits der Adreß-BUS von dieser Dekodierschaltung restlos dekodiert.
Wie aus Bild 3-14 ersichtlich, hat PIO Nr. 1 IOQ1, BIORQ, BA0 und BA1 als Eingänge. Ohne beim PIO-IC zu sehr ins Detail zu gehen (damit befaßt sich später ein ganzes Kapitel), soll es Ihnen kurz vorgestellt und auf die Bedeutung dieser Eingänge hingewiesen werden.
Die genaue Bezeichnung für das PIO-IC lautet "Z-80 Parallel Input/Output Interface Controller (Parallele Ein-/Ausgang-Interface-Steuerung). Dieses IC ist speziell für das Interfacing der Z-80-CPU mit den externen Geräten konzipiert. Die Bezeichnung "parallel I/O" bedeutet, daß vom PIO gleichzeitig 8 Bits zum externen Gerät gelangen.
Das PIO-IC ist programmierbar. *Programmierbar* bedeutet, die Arbeitsweise des IC wird mit Hilfe von Steuer-Bytes von der CPU genau festgelegt. Form, Bedeutung und der Inhalt der Steuer-Bytes sind in einem späteren Kapitel näher erläutert. Im Augenblick nur soviel, die Z-80-CPU kann programmiert werden, um geeignete Steuer-Bytes an das PIO-IC auszugeben, damit es auf jede von mehreren Arten operieren kann.

Der PIO-Chip hat zwei verschiedene I/O-Gatter. Dadurch kann ein Gatter als Ausgangs-Gatter an einen Drucker und das andere als Eingang an ein Tastenfeld angeschlossen werden. Die Bezeichnung der beiden Gatter für jedes Chip sind A und B. Jedes Gatter ist über seine eigenen Steuer-Bytes unabhängig ansprechbar. Das PIO-IC hat zwei Steuer-Gatter, eins für GATTER A und eins für GATTER B. Insgesamt ist das PIO-IC mit zwei Gattern ausgestattet (Gatter A, Gatter B und die dazugehörigen Steuer-Gatter). Wenn die Z-80-CPU sich mit dem PIO verbindet (Interfacing), ist das gewünschte Gatter zu bestimmen. Deshalb werden die beiden niedrigstwertigen Adressenleitungen vom PIO-IC in der in Bild 3-14 dargestellten Schaltung dekodiert, wobei eine Leitung das C/D- (Steuer- oder Daten-) Gatter und die andere das A/B-Gatter auswählt. Auch muß das PIO-IC für Ein-/Ausgabeoperationen aktiviert werden, für die ihre Gatter

adressiert sind. In Bild 3-14 übernehmen diese Aufgabe die Signale $\overline{BIORQ}$ und $\overline{IOQ1}$. Die Signale $\overline{BIORQ}$, $\overline{IOQ1}$, BA0 und BA1 haben folgende Funktionen:

$\overline{BIORQ}$: Deutet auf eine augenblickliche Ein-/Ausgabeoperation hin, eine Voraussetzung für die Aktivierung des PIO.

$\overline{IOQ1}$: Dieses Signal ist mit dem Freigabe-Gatter $\overline{CE}$ (chip enable) am PIO-Pin 1 verbunden. $\overline{IOQ1}$ ist nur dann aktiv, wenn sich BA2 im Zustand logisch 1 und BA3 bis BA7 im Zustand logisch 0 befinden. $\overline{IOQ1}$ dekodiert die oberen sechs Adressenleitungen um zu bestimmen, welcher PIO aktiviert werden soll.

BA0 und BA1 bestimmen die Adressierung der vier Gatter: GATTER A, Steuer-GATTER A, GATTER B oder Steuer-GATTER B.

BA0 und BA1: Diese Signale bestimmen, welches Gatter des von den Leitungen BA2 bis BA7 adressierten PIO-ICs während dieser Operation aktiviert sein soll. Gatter A oder B wird von der BA0-Leitung angewählt, BA0 im Zustand logisch 0 wählt Gatter B an. Die Steuer- oder Daten-I/O wählt die BA1-Leitung an; im Zustand logisch 1 ist die die Steuerung-I/O aktiviert.

Der Eingang von PIO Nr. 2 sind die $\overline{BIORQ}$-, $\overline{IOQ2}$-, BA0- und BA1-Anschlüsse. Somit wird PIO Nr. 2 nur dann aktiviert, wenn sich BA3 im Zustand logisch 1 und BA2, BA4, BA5, BA6 und BA7 im Zustand logisch 0 befinden. Die IOQ_n-Ausgänge des Dekoder/Demuliplexers 74LS138 aktivieren höchstens ein PIO-IC für irgendeine Ein-/Ausgabeoperation. Die Funktionen der $\overline{BIORQ}$-, BA0- und BA1-Signale sind bei PIO Nr. 1 und PIO Nr. 2 identisch.

Die Gatter-Adressen für die beiden mit dem Nanocomputer® verbundenen PIO-ICs sind folgende:

Gerätecode	Aktives PIO-IC	Gatter des aktiven PIO-IC
04	PIO Nr. 1	Daten-GATTER A
05	PIO Nr. 1	Daten-GATTER B
06	PIO Nr. 1	Steuer-GATTER A
07	PIO Nr. 1	Steuer-GATTER B
08	PIO Nr. 2	Daten-GATTER C
09	PIO Nr. 2	Daten-GATTER D
0A	PIO Nr. 2	Steuer-GATTER C
0B	PIO Nr. 2	Steuer-GATTER D

Beachten Sie die Umbenennungen der beiden Gatter auf dem PIO Nr. 2 gegenüber den Signal-Ausgängen der Experimentier-Platine: GATTER C (anstelle von GATTER A) und GATTER D (ANSTATT GATTER B).

Wie unterscheidet nun das PIO-IC zwischen den Signalen $\overline{WR}$ und $\overline{RD}$, d.h. zwischen den Ein- und Ausgabeoperationen? Diese Frage wird in dem Kapitel über das PIO-IC eingehend behandelt.

Eine ähnliche Schaltung wie in Bild 3-14 benutzt der Nanocomputer® NBZ80, um die I/O-Operationen mit zwei PIO-ICs auszuführen. Das $\overline{IOQ0}$-Signal steuert bei kompletten Mikrocomputern CLZ-80 einen

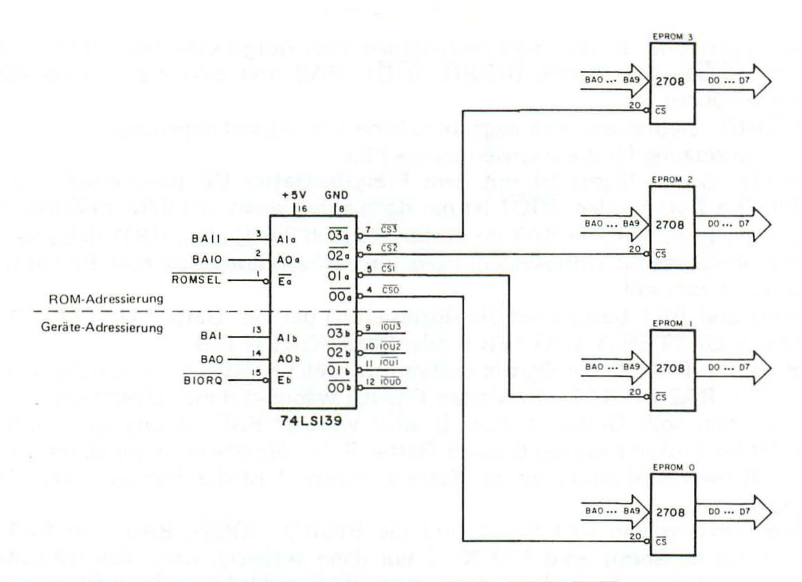


Bild 3-15. Schaltung zur Erzeugung der Signale $\overline{\text{IOU0}}$, $\overline{\text{IOU1}}$, $\overline{\text{IOU3}}$ und zur Dekodierung von Adressen in den höchstwertigen Speicher-Bytes 4K.

8251 USART. Dieses Intel-IC ist ein Unveral Synchronous/Asynchronous Receiver/Transmitter (Universal Synchron/Asynchron Empfänger/Sender). Die übrigen Signale $\overline{\text{IOQ3}}$, $\overline{\text{IOE0}}$, $\overline{\text{IOE1}}$, $\overline{\text{IOE2}}$ und $\overline{\text{IOE3}}$ stehen zur freien Verfügung. Die Schaltung in Bild 3-15 erzeugt weitere Dekodiersignale. Es sind dies:

$\overline{\text{IOU0}}$, $\overline{\text{IOU1}}$, $\overline{\text{IOU2}}$ und $\overline{\text{IOU3}}$

Diese Signale und vier Speicheradressen-Auswahlimpulse erzeugt das Dekoder/Demultiplexer-IC 74LS139.
Beispiel 3 hat praktische Nanocomputer®-Schaltungen beschrieben. Den einzelnen BUS-Bezeichnungen ist an der Z-80-CPU ein "B" vorangestellt. Das bedeutet: Der Ausgangslastfaktor der CPU ist so gering, daß die Signale selbst für die Steuerung aller angeschlossenen Schaltungen zu wenig Leistung aufweisen. Es muß also zwischen der CPU und der zu steuernden Schaltung eine Pufferstufe vorgesehen werden. In der Mikrocomputer-Literatur ist die Kennzeichnung der zu puffernden Signale mit dem Buchstaben B üblich (B: Buffer = Pufferstufe).

Adressen-Auswahlimpulse

Die Schaltung zur Erzeugung von Adressen-Auswahlimpulsen ist ähnlich wie die zur Erzeugung von Geräte-Auswahlimpulsen. Sie unterscheiden sich jedoch in zwei wesentlichen Dingen:

1. Anstatt 8 werden 16 Auswahlimpulse dekodiert. Als Auslöseimpuls der Dekodierschaltung ersetzt $\overline{\text{MREQ}}$ das $\overline{\text{IORQ}}$-Signal.

2. Die Speicher-ICs selbst führen viele der Adressen-Dekodierungen intern aus.

Das folgende Beispiel beschreibt die Dekodier-Schaltung des Nanocomputers® zum Dekodieren von Adressen in den 4K-Bytes des EPROM-Speichers von F000 bis einschließlich FFFF. Der Speicher besteht aus vier EPROMs 2708, die folgendermaßen belegt sind:

Adressenbereich	Entsprechender Anschluß
EPROM Nr. 1: F000-F3FF	Q49 (LHS)
EPROM Nr. 2: F400-F7FF	Q50
EPROM Nr. 3: F800-FBFF	Q51
EPROM Nr. 4: FC00-FFFF	Q52 (RHS)

Schreibt man die obigen Adressen in den Binärcode um, kann man erkennen, wie ausschließlich die Bits A10 und A11 den Adressenbereich kennzeichnen.

	BA15				BA0
EPROM Nr. 1	1111	00	00	0000	0000
	1111	00	11	1111	1111
EPROM Nr. 2	1111	01	00	0000	0000
	1111	01	11	1111	1111
EPROM Nr. 3	1111	10	00	0000	0000
	1111	10	11	1111	1111
EPROM Nr. 4	1111	11	00	0000	0000
	1111	11	11	1111	1111

Bei der Adresse FCCC = 1111 1100 1100 1100 ist sowohl BA10 als auch BA11 logisch 1; sie befindet sich daher im EPROM Nr. 4; die Adresse F503 = 1111 0101 0000 0011 ist bei BA10 logisch 1 und bei BA11 logisch 0; sie gehört ins EPROM Nr. 2. Der folgende Abschnitt beschreibt, wie der Nanocomputer® diese Information benutzt, um eine 16-Bit-Speicheradresse auf dem Adreß-BUS zu dekodieren und den entsprechenden EPROM 2708 zum geeigneten Zeitpunkt zu aktivieren.

Beispiel 1: Adressen-Auswahlimpulse für den EPROM-Speicher des Nanocomputers®

Die Schaltung in Bild 3-15 verwendet ein zweifaches 1-zu-4 Dekoder/Demultiplexer-IC 74LS139. Dieses IC verfügt je Dekoder über einen Strobe-Eingang, 2 Select-Eingänge und 4 Datenausgänge (Strobe = abtasten; Select = auswählen).

Informationen über den 74LS139 können Sie der Tabelle 3-5 entnehmen.

Bild 3-15 zeigt, wie der Nanocomputer® sich die beiden Dekoder/Demultiplexer im IC zu Nutze macht:

– Der Dekoder/Demultiplexer "a" wählt einen der vier EPROMs 2708 aus;
– Der Dekoder/Demultiplexer "b" erzeugt die Signale $\overline{\text{IOU0}}$, $\overline{\text{IOU1}}$, $\overline{\text{IOU2}}$ und $\overline{\text{IOU3}}$.

Das $\overline{\text{ROMSEL}}$-Signal (ROM SELect – ROM-Auswahl), das den Dekoder "a" aktiviert, wird von der in Bild 3-16 dargestellten Schaltung erzeugt.

Tabelle 3-5. Kenndaten und Eigenschaften des T54LS139/74LS139.

DUAL 1-OF-4 DECODER

DESCRIPTION – The LSTTL/MSI T54LS139/T74LS139 is a high speed Dual 1-of-4 Decoder/Demultiplexer. The device has two independent decoders, each accepting two inputs and providing four mutually exclusive active LOW outputs. Each decoder has an active LOW Enable input which can be used as a data input for a 4-output demultiplexer. Each half of the LS139 can be used as a function generator providing all four minterms of two variables. The LS139 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

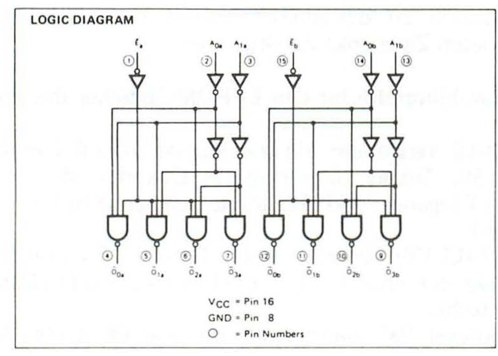
- SCHOTTKY PROCESS FOR HIGH SPEED
- MULTIFUNCTION CAPABILITY
- TWO COMPLETELY INDEPENDENT 1-OF-4 DECODERS
- ACTIVE LOW MUTUALLY EXCLUSIVE OUTPUTS
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

PIN NAMES

		LOADING (Note a)	
		HIGH	LOW
A_0, A_1	Address Inputs	0.5 U.L.	0.25 U.L.
$\bar{E}$	Enable (Active LOW) Input	0.5 U.L.	0.25 U.L.
$\bar{O}_0 - \bar{O}_3$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

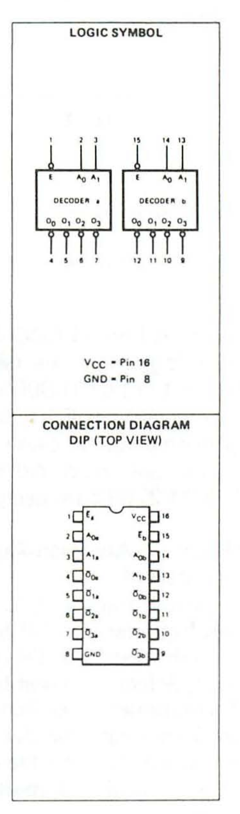
NOTES:
a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



LOGIC SYMBOL

CONNECTION DIAGRAM DIP (TOP VIEW)



FUNCTIONAL DESCRIPTION – The LS139 is a high speed dual 1-of-4 decoder/demultiplexer fabricated with the Schottky barrier diode process. The device has two independent decoders, each of which accept two binary weighted inputs (A_0, A_1) and provide four mutually exclusive active LOW outputs ($\overline{O}_0$-$\overline{O}_3$). Each decoder has an active LOW Enable ($\overline{E}$). When $\overline{E}$ is HIGH all outputs are forced HIGH. The enable can be used as the data input for a 4-output demultiplexer application.

Each half of the LS139 generates all four minterms of two variables. These four minterms are useful in some applications, replacing multiple gate functions as shown in Fig. a, and thereby reducing the number of packages required in a logic network.

TRUTH TABLE

INPUTS			OUTPUTS			
$\overline{E}$	A_0	A_1	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

Fig. a

Eine vereinfachte, aber logisch äquivalente Definition von $\overline{\text{ROMSEL}}$ ist folgende:

$$\overline{\text{ROMSEL}} = \overline{A12 \cdot A13 \cdot A14 \cdot A\,15 \cdot \overline{\text{MREQ}} \cdot \overline{\text{RFSH}}}$$

Die schaltungstechnische Realisierung dieser Gleichung ist mit einem NAND-Gatter möglich (Bild 3-17).

$\overline{\text{ROMSEL}}$ ist nur dann aktiv, wenn sich A12, A13, A14 und A15 alle im Zustand logisch 1 befinden, $\overline{\text{MREQ}}$ ist aktiv (bei logisch 0) und $\overline{\text{RFSH}}$ ist nicht aktiv. Dies ist verständlich, denn $\overline{\text{ROMSEL}}$ ist das NAND von PARGO und $\overline{\text{RFSH}}$. Wenn $\overline{\text{ROMSEL}}$ aktiv sein soll (low), darf $\overline{\text{RFSH}}$ nicht aktiv sein (d.h. es muß im Zustand logisch 1 sein) und PARGO muß aktiv (logisch 1) sein.

Das Signal $\overline{\text{RFSH}}$ verhindert den Zugriff zu einem der EPROMs, wenn ein Speicher-Refreshzyklus im Gange ist. Durch Zerlegen des $\overline{\text{RFSH}}$-Signals in Dekoder-Logikfaktoren werden die Leitungen "klar" gemacht, und zwar vor jeglichem Versuch, EPROM-Daten abzulesen. In Kapitel 4 werden Sie erfahren, warum der BUS klar sein muß.

Das PARGO-Signal ist unter bestimmten Vorausstetzungen, logisch 1 (siehe Bild 3-16).

PARGO = "1", aber nur, wenn beide Eingänge zum AND-Gatter 74LS08 logisch 1 sind;

PARGO = "1", aber nur, wenn der Ausagng vom NOR-Gatter 74LS02 logisch 1 ist;

PARGO = "1", aber nur, wenn beide Eingänge zum NOR-Gatter 74LS302 logisch 0 sind;

PARGO = "1", aber nur, wenn die 74LS139-Ausgänge $\overline{\text{03a}}$ und $\overline{\text{03b}}$ logisch 0 sind.

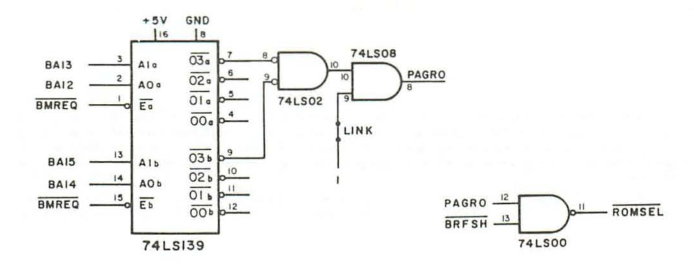


Bild 3-16. Nanocomputer®-Schaltung zur Erzeugung der Auswahlimpulse $\overline{\text{ROMSEL}}$ und PARGO ($\overline{\text{BRFSH}}$ ist das gepufferte $\overline{\text{RFSH}}$-Signal der Z-80-CPU).

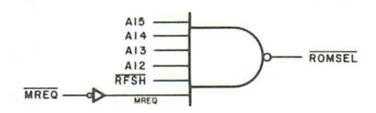


Bild 3-17. Einzeldarstellung eines logischen Gatters für die Dekodierungsschaltung (Bild 3-16), die das $\overline{\text{ROMSEL}}$-Signal erzeugt.

03a = "0", aber nur, wenn $\overline{\text{BMREQ}}$ aktiv ist UND (BA12 = "1") UND (BA13 = "1");

03b = "0", aber nur, wenn $\overline{\text{BMREQ}}$ aktiv ist UND (BA14 = "1") UND (BA15 = "1").

PARGO = "1", aber nur, wenn $\overline{\text{BMREQ}}$ = "0", BA12 = "1", BA13 = 1, BA14 = "1" UND BA15 = 1 sind.

$\overline{\text{BMREQ}}$ ist "0" und somit äquivalent zu BMREQ = 1,

$$\text{PARGO} = \text{BA12} \cdot \text{BA13} \cdot \text{BA14} \cdot \text{BA15} \cdot \text{BMREQ}$$

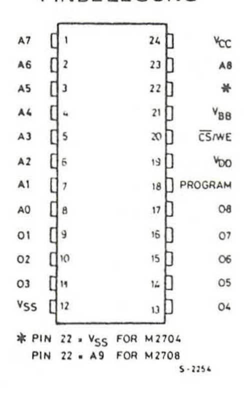
und folglich

$$\overline{\text{ROMSEL}} = \overline{\text{BA12} \cdot \text{BA13} \cdot \text{BA14} \cdot \text{BA15} \cdot \text{BMREQ} \cdot \overline{\text{BRFSH}}}$$

Somit ist der Dekoder/Demultiplexer nur dann aktiv, wenn eine Speicher-Zugriffoperation im Gange ist (außer Speicher-Refresh) und die Adresse auf dem Adreß-BUS in den Bereich F000 . . . FFFF fällt. Die einzige noch zu treffende Entscheidung ist die, welcher EPROM aktiviert werden soll. In Bild 3-15 geschieht dies mit Hilfe der BA10- und BA11-Signale; sie wählen einen der vier 74LS319-Ausgänge $\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS2}}$ oder $\overline{\text{CS3}}$ an. Jedes CSn-(Chip Select n = IC-Auswahl n) Signal ist mit dem EPROM verbunden, der zu dem entsprechenden BA10- und BA11-Dekoder gehört.
Bisher sind die Bits A10 bis einschließlich A15 der Adressenleitungen bei der Adressierung eines EPROM-Bytes beschrieben. Wie steht es mit A0 bis einschließlich A9? Die Antwort lautet, A0 bis A9 sind mit den Eingängen eines jeden der vier EPROMs verbunden (in Bild 3-15 BA0 . . . 9). Jedes 2708-EPROM kann genau 1024 oder 2^{10} Bytes aufnehmen. Die Signale auf den 10 Adreßleitungen (A0 bis A9) bestimmen das Byte, welches vom aktivierten EPROM auf den Daten-BUS gelegt werden soll. Das EPROM selbst kombiniert eins seiner gespeicherten Bytes mit der

Adresse auf den 10 Eingangsleitungen und legt das Byte auf seine acht Daten-Ausgangsleitungen. In Bild 3-18 ist ein Blockschaltbild und eine Pinbelegungsskizze für das EPROM 2708 dargestellt.

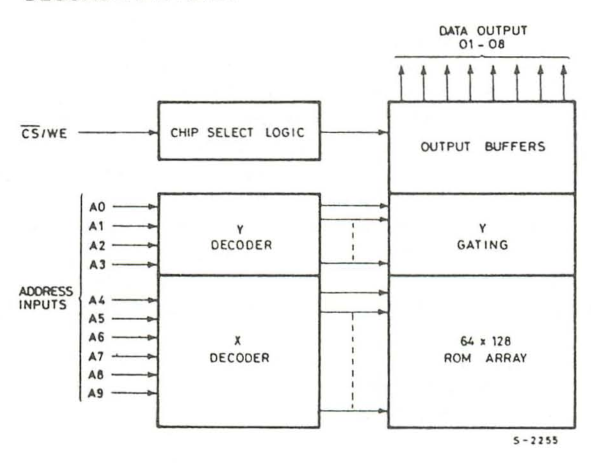
PINBELEGUNG



PINBEZEICHNUNG

A_0-A_9	ADRESSEN-EINGÄNGE
O_I-O_8	DATEN-AUSGÄNGE
$\overline{CS}$/WE	(CHIP-AUSWAHL/SCHREIB-FREIGABE-EINGANG)

BLOCKSCHALTBILD



MODE	ANSCHLUßPIN						
	9,11,13,17	12	18	19	20	21	24
READ	D_{OUT}	V_{SS}	V_{SS}	V_{DD}	V_{IL}	V_{BB}	V_{CC}
PROGRAM	D_{IN}	V_{SS}	Pulsed V_{IHP}	V_{DD}	V_{IHW}	V_{BB}	V_{CC}

Bild 3-18. Blockschaltbild und Pinbelegung des EPROM 2708.

Beispiel 2: Adressen-Auswahlimpulse für den RAM-Speicher des Nanocomputers®

Die folgende Schaltung ist nach relativ modernen Gesichtspunkten konzipiert. Wenn Sie beim Lesen nicht alles auf Anhieb verstehen, ist das kein Grund zur Besorgnis. Wiederholen Sie den entsprechenden Abschnitt zu gegebener Zeit.

In Bild 3-19 ist ein vereinfachtes Schaltbild der Schaltung dargestellt, die der Nanocomputer® zum Dekodieren von RAM-Adressen-Auswahlimpulsen verwendet. Der Standard-RAM-Speicher für den Nanocomputer® besteht aus acht dynamischen 4 KilobitRAM-ICs 4027, die im Bereich 0000 . . . 0FFF (hex) adressiert sind. Man kann den Speicherort durch Verstellen der Weichen auf der PC-Platine des Nanocomputers® variieren. Für dieses Beispiel sollen die Weichen den Speicherort im Bereich 0000 . . . 0FFF (hex) lokalisieren. Das RAM des Nanocomputers® ist physikalisch so aufgebaut, daß jedes Chip ein Bit für jedes der gespeicherten 4 Kilobytes liefert. Um allein die 4 Kilobytes in jedem RAM-Chip zu adressieren, sind 12 Adressenleitungen erforderlich.

Anstatt 12 Adreß-Eingangspins hat der Hersteller bei den RAM-ICs nur sechs Adreß-Eingangspins vorgesehen. Das RAM nimmt daher die Adreßsignale in zwei Blöcke zu je sechs Bits auf. Dafür sind zwei Auswahlimpulse erforderlich: RAS für A0 . . . A5 und CAS für A6 . . . A11 (siehe Pinbelegung für das RAM 4027 in Bild 3-20). RAS bedeutet Row Adress Select (Reihen-Adressenauswahl) und CAS heißt Column Adress Select (Spalten-Adressenauswahl).

Zwei Anschlußstifte sind mits $\overline{\text{RAS}}$ und $\overline{\text{CAS}}$ bezeichnet! Wenn das $\overline{\text{RAS}}$-Signal aktiv (low bzw. logisch 0) ist, wird die 6-Bit-Adresse auf den Leitungen MA0 . . . MA5 als Reihen-Adresse übersetzt. Ein aktives $\overline{\text{CAS}}$-Signal (ebenfalls low) übersetzt die Adresse als eine Spalten-Adresse. Der Vorgang, sechs Adressen-Eingänge für eine 12-Bit-Adresse zu je 2 x 6 Bits zu benutzen, wird *Adressen-Multiplexierung* genannt. Hierfür sind innerhalb und außerhalb des ICs ausgereifte logische Konzepte in den Schaltungen zur CPU erforderlich. Die entsprechende Schaltung des Nanocomputers® ist im Anhang dargestellt. Bild 3-19 zeigt die wichtigen Signale und faßt die Verzögerungsschaltung zur Erzeugung der $\overline{\text{SELAD}}$- und $\overline{\text{CAS}}$-Signale mit einem "Funktionskasten" zusammen.

Bei einem Speicherlese- oder-schreibzyklus laufen folgende Funktionen ab:

1. Die Adresse der Speicherstelle legt die CPU auf den Adres-BUS BA0 . . . BA15.
2. BA12, BA13, BA14 und BA15 sind Eingaben für einen Dual-Dekoder/Demultiplexer mit 2 bis 4 Leitungen.
3. Das $\overline{\text{BMREQ}}$-Signal, das beide Dekoder/Demultiplexer des ICs 74LS139 freigibt, wird aktiviert.
4. Befinden sich BA12, BA13, BA14 und BA15 im Zustand logisch 0, ist die Adresse für eine Speicherstelle im Bereich 0000 . . . 0FFF bestimmt. Die dynamischen RAMs werden durch Aktivierung des Signals PAGRA (PAGe of RAM – Seite des RAM) ausgewählt. In Bild 3-19 ist PAGRA aktiv (high), aber nur, wenn 00a und 00b auf dem Dekoder/Demultiplexer aktiviert sind. Folglich ist
   PAGRA nur dann high-aktiv, wenn $\overline{\text{BMREQ}}$ aktiv sowie BA12 = 0, BA13 = 0, BA14 = 0 und BA15 = 0 sind.

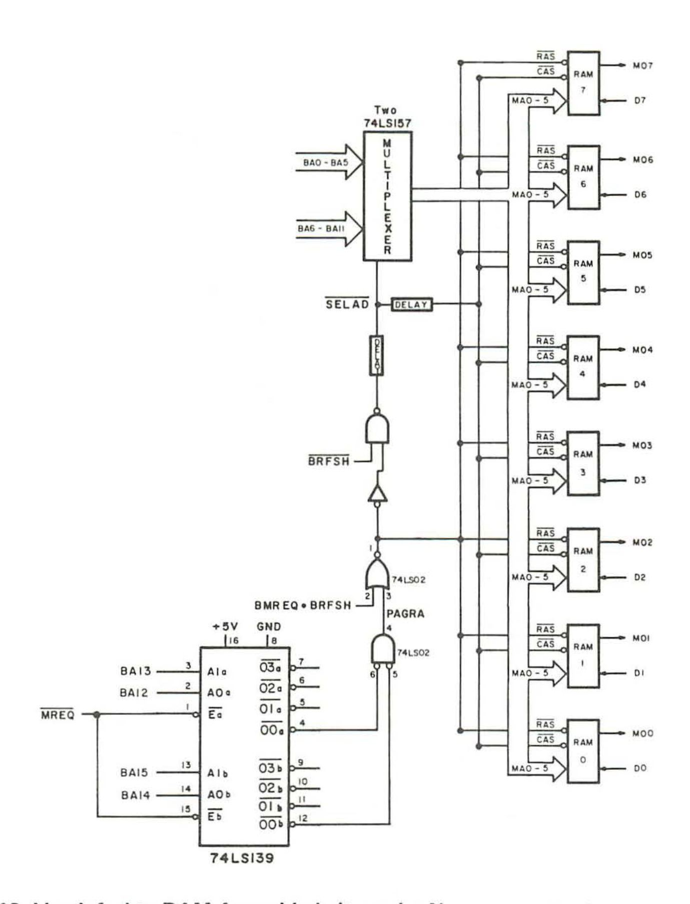


Bild 3-19. Vereinfachte RAM-Auswahlschaltung des Nanocomputers®.

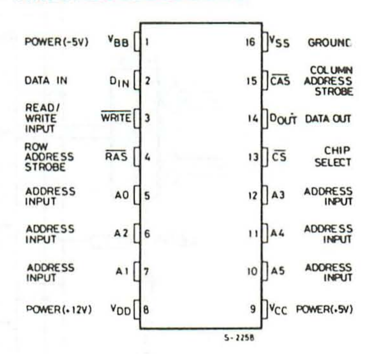
Da also BMREQ nur dann low-aktiv ist, wenn BMREQ high ist, gilt:

$$PAGRA = BMREQ \cdot \overline{BA12} \cdot \overline{BA13} \cdot \overline{BA14} \cdot \overline{BA15}$$

Der 4K-RAM-Speicher kann auch an anderen Adressen lokalisiert werden, indem man verschiedene Ausgänge des ICs 74LS139 wählt, um das PAGRA-Signal zu aktivieren.

5. Das PAGRA-Signal wird Pin 3 eines 74LS02-NOR-Gatters zugeführt. Ist PAGRA logisch 1, wechselt das Ausgangssignal des 74LS02 ($\overline{RAS}$) unabhängig vom Logikpegel am anderen Gattereingang auf logisch 0.
6. Die Aktivierung des $\overline{RAS}$-Signals veranlaßt die RAM-ICs 4027 dazu, die Adresse auf den Leitungen MA0 . . . MA5 abzulesen und als Speicherzellen-Reihenadresse zu übersetzen. Welche Adresse liegt auf den MA0 . . . MA15-Leitungen bei Aktivierung von RAS? Die Antwort auf diese Frage basiert auf der Funktion der beiden 74LS157-Multiplexer-ICs in der Schaltung Bild 3-19. Die Pinbelegung des 74LS157 geht aus Bild 3-21 hervor.

ANSCHLUßBELEGUNG



BLOCKSCHALTBILD

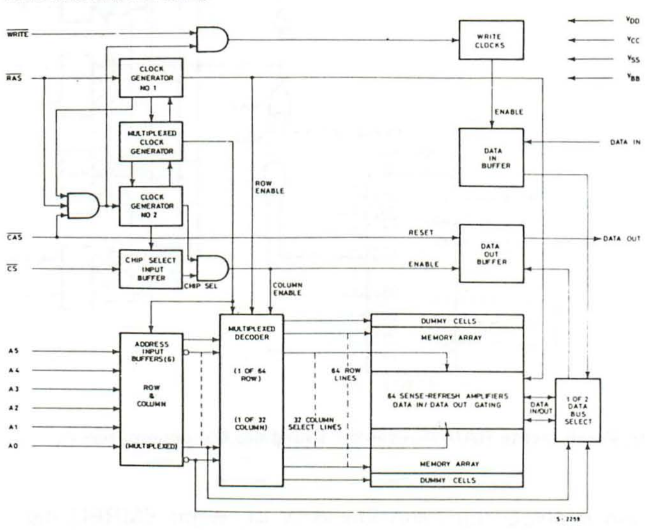


Bild 3-20. Pinbelegung und Blockschaltbild des dynamischen RAM-ICs 4027.

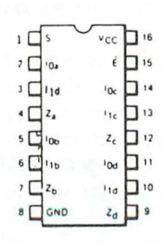


Bild 3-21. Pinbelegung des ICs 74LS157.

Ist das SELAD-Signal inaktiv (high), werden die Adreß-Leitungen BA0 . . . BA5 zu den Ausgängen des Multiplexers geschaltet, die mit MA0 . . . MA5 verbunden sind. Da zwischen der Aktivierung von RAS und SELAD eine Verzögerung liegt, befindet sich die niederwertigere Hälfte der 12-Bit-Speicheradresse auf MA0 . . . MA5, nachdem RAS aktiviert ist. RAS tastet also BA0 . . . BA5 als untere Adresse in die RAMs 4027.

7. Ist das RAS-Signal aktiviert (log. 0), wird nach einer Verzögerung SELAD ebenfalls logisch 0. Der Multiplexer legt BA6 . . . BA11 auf seine Ausgangsleitungen und stabilisiert sie nach einer weiteren Verzögerung auf MA0 . . . MA5. Daraufhin wird das CAS-Signal aktiviert (logisch 0).
8. Die höherwertige Hälfte der 12-Bit-Speicheradresse BA11 . . . BA16 gelangt als Spaltenadresse in die RAMs 4027.

Das BRFSH-Signal spielt bei Speicher-, Lese- und Schreiboberationen eine untergeordnete Rolle; es ist deshalb in der obigen Funktionsbeschreibung nicht erwähnt. Warum erscheint das BRFSH-Signal denn in dieser Schaltung? Um diese Frage beantworten zu können, muß eine wichtige Eigenschalt des dynamischen RAM klar sein, nämlich die Notwendigkeit für periodisches *Refreshing.*

Dynamische RAMs speichern Bits auf der Basis von geladener Kapazität antatt mit Flipflops. Daher ist es nötig, den Speicher periodisch "aufzufrischen", bevor sich die Kapapzität entladen kann. Die Z-80-CPU stellt das dynamische RAM-Refresh-Signal RFSH in Form einer Refresh-Adresse und eines Refresh-Abtastimpulses zur Verfügung. RFSH ist gepuffert und erscheint auf dem BUS des Nanocomputers® als BRFSH. Bei der Refresh-Adresse handelt es sich um eine Reihen- oder Spalten-Auswahladresse, die im R-Register im Innern der Z-80-CPU gespeichert ist. Am Ende eines jeden M1-Zyklus (Objektcode-Abruf) erzeugt die CPU diese Refresh-Signale. In Bild 3-22 ist der M1-Zyklus einer Z-80-CPU dargestellt. Gleichzeitig mit der Aktivierung des RFSH-Signals wird der Inhalt des R-Registers (die Refresh-Adresse) auf die niederwertigere Hälfte des Adreß-BUS gelegt. In der Schaltung Bild 3-19 frischen folgende Ereignisse die dynamischen RAMs auf:

1. Die Aktivierung des BRFSH-Signals sperrt die Signale SELAD und CAS, weil SELAD zwangsläufig auf logisch 1 geht.
2. Die ersten sechs Bits der Refresh-Adresse auf BA0 . . . BA5 werden dann durch die Multiplexer-ICs 74LS157 zu den MA0 . . . MA5-RAM-Eingängen blockiert.
3. Das RAS-Signal wird aktiviert, weil auch das Signal BMREQ · BRFSH an Pin 2 des 74LS02 aktiviert ist (logisch 1). In diesem Fall ist der logische Zustand des PAGRA-Signals ohne Bedeutung.
4. Mit der Aktivierung von RAS wird die Refresh-Adresse in die RAM-ICs als Reihen-Auswahladresse getaktet. Da CAS in diesem Fall nicht freigegeben wird, benutzt das RAM die Refresh-Adresse, um alle Zellen in der festgelegten Reihe aufzuladen. Jeder folgende M1-Zyklus erhöht das Refresh-Register R.

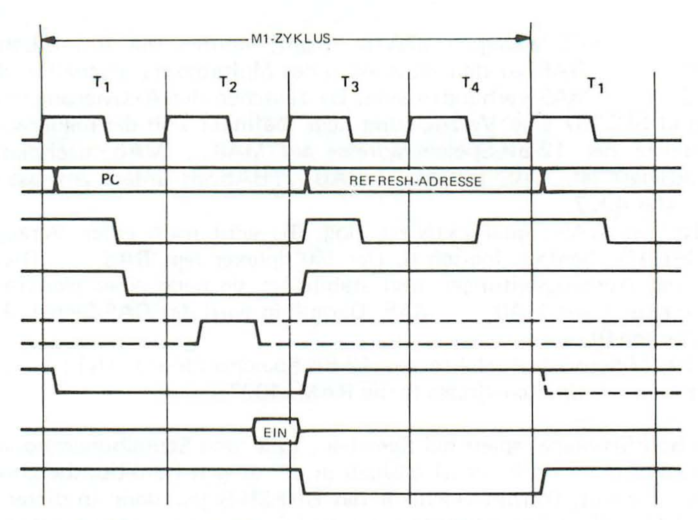


Bild 3-22. M1-Zyklus der Z-80-CPU.

Die Auffrischung aller Reihen im gesamten RAM von 4k x 8 Bits geschieht in 128 aufeinanderfolgenden M1-Zyklen.

ZUSAMMENFASSUNG ALLER I/O-AUSWAHLSIGNALE

Wie bereits erwähnt, gibt es neun I/O-Auswahlsignale: $\overline{IOQ3}$, $\overline{IOE0}$, $\overline{IOE1}$, $\overline{IOE2}$, $\overline{IOE3}$, $\overline{IOU0}$, $\overline{IOU1}$, $\overline{IOU2}$ und $\overline{IOU3}$. Sie stehen dem Anschluß B auf der Experimentier-Platine des Nanocomputers® zur Verfügung. Der folgende Abschnitt definiert die einzelnen Signale und beschreibt die Verwendung in I/O-Schaltungen, deren niederwertigere acht Bits des Adreß-BUS mehrdeutig dekodiert werden müssen.
Auf der PC-Platine des Nanocomputers® werden insgesamt 12 I/O-Auswahlsingale erzeugt: die neun für die Experimentier-Platine plus $\overline{IOQ0}$, $\overline{IOQ1}$ und $\overline{IOQ2}$. Die letzten drei Signale sind ausschließlich dem Betriebssystem des Nanocomputers® vorbehalten und stehen dem Benutzer daher nicht zur Verfügung. Der Vollständigkeit halber sind nachstehend alle 12 Signale definiert:

$$\overline{IOU0} = \overline{BIORQ} \cdot \overline{BA0} \cdot \overline{BA1}$$
$$\overline{IOU1} = \overline{BIORQ} \cdot \overline{BA0} \cdot BA1$$
$$\overline{IOU2} = \overline{BIORQ} \cdot BA0 \cdot BA1$$
$$\overline{IOU3} = \overline{BIORQ} \cdot BA0 \cdot BA1$$
$$\overline{IOQ0} = \overline{BA2} \cdot \overline{BA3} \cdot \overline{BA4} \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}$$
$$\overline{IOQ1} = \overline{BA2} \cdot \overline{BA3} \cdot \overline{BA4} \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}$$
$$\overline{IOQ2} = BA2 \cdot \overline{BA3} \cdot \overline{BA4} \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}$$
$$\overline{IOQ3} = BA2 \cdot BA3 \cdot \overline{BA4} \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}$$
$$\overline{IOE0} = \overline{BA2} \cdot \overline{BA3} \cdot BA4 \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}$$
$$\overline{IOE1} = \overline{BA2} \cdot \overline{BA3} \cdot BA4 \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}$$
$$\overline{IOE2} = \overline{BA2} \cdot BA3 \cdot BA4 \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}$$
$$\overline{IOE3} = BA2 \cdot BA3 \cdot BA4 \cdot \overline{BA5} \cdot \overline{BA6} \cdot \overline{BA7}$$

Die Bilder 3-14 und 3-15 zeigen die Dekoder für diese Signale. Bei den Bezeichnungen handelt es sich die gepufferten Signale (BA0 . . . BA15), wie sie auf dem BUS des Nanocomputers® zur Verfügung stehen.

Durch eine paarweise Kombination der obigen Signale entweder mit BRD, $\overline{\text{BWR}}$ oder $\overline{\text{DBWR}}$, IN n und OUT n können Geräte-Auswahlimpulse erzeugt werden, wobei n die hexadezimale Zahl im Bereich 00 . . . 1F ist. Tabelle 3-6 macht den Zusammenhang zwischen den hexadezimalen Gerätecodes (n) und den entsprechenden Signalpaaren deutlich. Um z.B. den Geräte-Auswahlimpuls OUT 11H zu erzeugen, suche man das Kästchen 11 in der Tabelle sowie die entsprechende Reihen- und Spaltenüberschrift (IOU1 und IOE0) und kombiniere $\overline{\text{BWR}}$ oder DBWR mit $\overline{\text{IOU1}}$ und $\overline{\text{IOE0}}$, wie in Bild 3-23 dargestellt.

$\overline{\text{IOU1}}$
$\overline{\text{IOEØ}}$
$\overline{\text{BWR}}$ or $\overline{\text{DBWR}}$
OUT 11H

Bild 3-23. Beispiel einer Kombination von Geräte-Auswahlsignalen in Tabelle 3-6.

Tabelle 3-6. Geräte Auswahlsignale des Nanocomputers® (die Signale $\overline{\text{IOU0}}$ bis $\overline{\text{IOU3}}$ werden von den Adreßleitungen A0 und A1 erzeugt, $\overline{\text{IOQ0}}$ bis $\overline{\text{IOE3}}$ von den Leitungen A2 bis A4 regeneriert.

I/O Adresse	$\overline{\text{IOU0}}$	$\overline{\text{IOU1}}$	$\overline{\text{IOU2}}$	$\overline{\text{IOU3}}$
$\overline{\text{IOQ0}}$	0 UART	1 UART	2 unbenutzt	3 I/O-Flag
$\overline{\text{IOQ1}}$	4 Daten-Gatter "A"	5 Daten-Gatter "B"	6 Steuer-Gatter "A"	7 Steuer-Gatter "B"
$\overline{\text{IOQ2}}$	8 Daten-Gatter "C"	9 Daten-Gatter "D"	A Steuer-Gatter "C"	B Steuer-Gatter "D"
$\overline{\text{IOQ3}}$	C	D	E	F
$\overline{\text{IOE0}}$	10	11	12	13
$\overline{\text{IOE1}}$	14	15	16	17
$\overline{\text{IOE2}}$	18	19	1A	1B
$\overline{\text{IOE3}}$	1C	1D	1E	1F

nicht für den externen Gebrauch verfügbar

verfügbare Leitungen

Aus Tabelle 3-6 ist auch ersichtlich, wie die Hardware des Nanocomputers® die Gatter 00 über 0B eingesetzt hat. Sowohl $\overline{\mathrm{IOQ1}}$ als auch $\overline{\mathrm{IOQ2}}$ kann man bequem als IC-Auswahl-Eingänge für die beiden PIO-ICs einsetzen, während der PIO selbst BA0 und BA1 direkt dekodiert (dadurch benötigt man zur PIO-Adressierung keine IOUn-Signale). Bei der Verdrahtung der PIO-Schaltung in einem nachfolgenden Versuch sind die Signale $\overline{\mathrm{IOQ3}}$ oder IOEn in gleicher Weise nützlich.

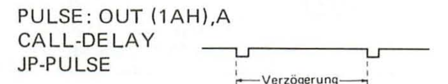
WEITERE ANWENDUNGSMÖGLICHKEITEN FÜR GERÄTE-AUSWAHLIMPULSE

Geräte-Auswahlimpulse sind relativ leicht zu erzeugen; sie ermöglichen einen Austauch zwischen Software und Hardware. Außer für Speicher-Zugriffoperationen und Geräte-I/O-Abtastfunktionen läßt sich der Geräte-Auswahlimpuls für viele Anwendungen einsetzen. In diesem Abschnitt werden einige der üblichen Anwendungen für Geräte-Auswahlimpulse behandelt; sie machen interessante Unterschiede zwischen Hardware und Software deutlich.

Beispiel 1: Anwendung des Geräte-Auswahlimpulses zur Erzeugung von Taktimpulsen
Es ist leicht, ein Programm zu schreiben, das einen einzigen negativen Impuls erzeugt:

```
OUT (1AH),A
```

Für eine Z-80-CPU, die mit 2,5 MHz arbeitet, beträgt die Impulsdauer (als AND von $\overline{\mathrm{IORQ}}$ und $\overline{\mathrm{WR}}$) ca. 3,25 T–Zyklen oder 1,625 Mikrosekunden. Für eine Folge dieser Impulse kann man ein einfaches Programm verwenden:



Die Subroutine DELAY ist eine programmierte Verzögerungsschleife, die in der gewünschten Zwischenimpulszeit operiert. Bei den erzeugten Impulsen handelt es sich um Geräte-Auswahlimpulse der Adresse 1AH. Will man die Impulsdauer ändern, ist das relativ leicht möglich. Dazu benötigt man ein Flipflop (z.B. 74LS74) und ein Paar der Geräte-Auswahlimpulse. Davon wird ein Impuls dem Setz-Eingang "Set" und ein Impuls dem Rücksetz-Eingang "Clear" zugeführt.
Mit zwei Zeitverzögerungsschleifen kann man eine Impulsfolge mit bekanntem Arbeitszyklus erzeugen (Bild 3-25).
Die letzten beiden Schaltungen ersetzen sowohl einen mono- als auch einen astabilen Multivibrator (wie z.B. den als Oszillator geschalteten Zeitgeber 555 IC).

Beispiel 2: Abtasten von integrierten Schaltungen
Eine wichtige Anwendung für Mikrocomputer ist das Abtasten von einzel-

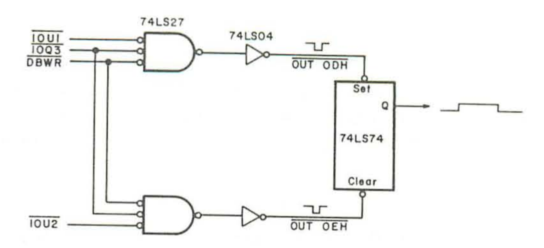


Bild 3-24. Erzeugung von Impulsen mit variabler Dauer.

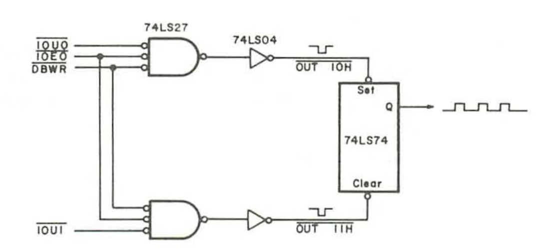


Bild 3-25. Erzeugung von Impulsfolgen.

nen integrierten Schaltungen in Instrumenten und elektronischen Geräten. Solche Impulse können zum Beispiel:

- Zähler, Schiebe-Register, Flipflops und Auffang-Register auf 0 stellen;
- Zähler, Schiebe- und Auffang-Register laden;
- Multiplexer, Demultiplexer, Dekoder, Zähler, Schiebe-Register, Speicher, "UARTS" (universele asynchrone Sender/Empfänger), PIO-ICs sowie eine Reihe anderer ICs freigeben;
- Takteingänge zu Zählern und Schiebe-Registern sperren;
- Flipflops setzen, rücksetzen und takten;
- Verschiebung links, Verschiebung rechts in Schiebe-Registern wählen, Schiebe-Register laden und die Funktionen sperren;
- das Programm anderer Digitalcomputer unterbrechen.

Die Verwendung von Geräte-Auswahlimpulsen zur Operationsteuerung von einzelnen integrierten Schaltungen ist eine Möglichkeit, Hardware gegen Software auszutauschen.

Oft ist es Ziel eines Schaltungskonzepts, die Zahl von ICs auf ein Minimum zu begrenzen. Die Verwendung von Mikrocomputern in Schaltungen ermöglicht dem Entwickler Funktionen, die normalerweise in Hardware verankert sind, gegen Software auszutauschen, was zu geringeren Zahlen führen kann. Während der Austausch von Software gegen Hardware oftmals erwünscht ist, gibt es Bedingungen, unter denen dies nicht möglich ist. Erstens: Während die Kosten für Hardware allgemein sinken, steigen sie für Software an. So ist es möglich, daß die Kosten für ein Programm als Ersatz für eine Hardware-Schaltung nicht niedriger sind.

Zweitens: Die Software ist gegenüber der Hardware oftmals langsamer, daher muß man alle Anforderungen an Geschwindigkeit und Zeit sorgfältig prüfen. Bei einem AND-Gatter kann die Signalverzögerung je nach Gattertyp 10 bis 100 Nanosekunden betragen, während ein AND-Befehl 4 T-Zyklen dauert (ca. 1,6 Mikrosekunden bei Ihrem Nanocomputer®).

RÜCKBLICK

Geben Sie für jeden der folgenden Befehle an, wieviele Objektcode-Abrufe (M1), Speicherlese-, Speicherschreib-, Geräte-Eingabe- und Geräte-Ausgabezyklen bei der Ausführung des entsprechenden Befehls stattfinden. Zum Beispiel braucht die CPU für den Befehl LD A,B lediglich einen Objektcode-Abrufzyklus auszuführen, während für den Befehl INC (IX) zwei Objektcode-Abrufzyklen (M1) erforderlich sind, weil er einen Zwei-Byte-Objektcode hat. Darüber hinaus muß die CPU noch folgende Zyklen ausführen: 1 Speicher-Eingabezyklus, um das Byte an der von IX bezeichneten Adresse einzugeben;
1 Speicher-Schreibzyklus, um das erhöhte Byte zu der von IX bezeichneten Adresse zurückzugeben.

Füllen Sie die nachstehende Tabelle – wie in den beiden Beispielen angegeben – fertig aus.

Befehl (Quell-Code)	M1-Zyklus	Speicher-Lese-zyklus	Speicher-Schreib-zyklus	Eingabe-zyklus	Ausgabe-zyklus
LD A,B	1	0	0	0	0
INC (IX)	2	1	1	0	0
LD A,00H					
LD C,D					
INC A					
DEC (HL)					
LD HL,00FFH					
LD (BC),A					
LD (0010H),A					
POP HL					
PUSH BC					
LD (0100H),1234H					
EXX					
XOR A					
XOR (HL)					
DAA					
DEC IX					
RRA					
RLC (IY+03)					
SET 3,(HL)					
DJNZ F4H					
JP (IX)					
CALL 0100H					
RET					
IN A,(04H)					
OUT (05H),A					
IN B,(C)					
OUT (C),D					

LÖSUNG

Befehl (Quell-Code)	M1-Zyklus	Speicher-Lese-zyklus	Speicher-Schreib-zyklus	Eingabe-zyklus zyklus	Ausgabe-zyklus
LD A,B	1	0	0	0	0
INC (IX)	2	1	1	0	0
LD A,00H	1	1	0	0	0
LD C,D	1	0	0	0	0
INC A	1	0	0	0	0
DEC (HL)	1	1	1	0	0
LD HL, 00FFH	1	2	0	0	0
LD (BC),A	1	0	1	0	0
LD (0010H),A	2	2	1	0	0
POP HL	1	2	0	0	0
PUSH BC	1	0	2	0	0
LD (0100H),1234H	2	2	2	0	0
EXX	1	0	0	0	0
XOR A	1	0	0	0	0
XOR (HL)	1	1	0	0	0
DAA	1	0	0	0	0
DEC IX	2	0	0	0	0
RRA	1	0	0	0	0
RLC (IY+03)	2	1	1	0	0
SET 3,(HL)	2	1	1	0	0
DJNZ F4H	1	1	0	0	0
JP (IX)	2	0	0	0	0
CALL 0100H	1	2	2	0	0
RET	1	2	0	0	0
IN A,(04H)	1	0	0	1	0
OUT (05H),A	1	0	0	0	1
IN B,(C)	1	0	0	1	0
OUT (C),D	1	0	0	0	1

EINFÜHRUNG IN DIE VERSUCHE

Die folgenden Versuche demonstrieren, wie Sie Geräte-Auswahlimpulse erzeugen und verwenden können. Sie verdrahten auch eine *Bit-Resonator*-Schaltung, die es Ihnen ermöglicht, einzelne Bits auf dem Adreß- und Daten-BUS zu beobachten, während die CPU Speicherzugriffs- bzw. Ein-/Ausgabebefehle ausführt.

Versuch-Nr.	Bemerkung
1	Demonstriert eine Bit-Resonator-Schaltung, die es ermöglicht, einzelne Bits auf dem Adreß- und Daten-BUS abzutasten. Sie benutzen diese Schaltung, um diese BUS-Leitungen bei der Ausführung eines Speicherzugriffs- oder Ein-/Ausgabe-Befehls zu beobachten.
2	Demonstriert eine Dekodierschaltung mit dem Dekoder/Demultiplexer-IC 74LS139.
3	Demonstriert die Verwendung eines Geräte-Auswahlimpulses zur Abtastung eines Dekadenzählers 74LS90.

4	Demonstriert eine Speicherschaltung, die zwei statische RAM-ICs 2101 von Intel (4 X 256) sowie ein AND-Gatter mit acht Eingängen zur Adressen-Dekodierung verwendet.
5	Demonstriert den Unterschied zwischen einem statischen und einem dynamischen RAM in bezug auf Auswirkungen der Wartezustände bei CPU-Speicher-Refresh-Operationen.

VERSUCH NR. 1

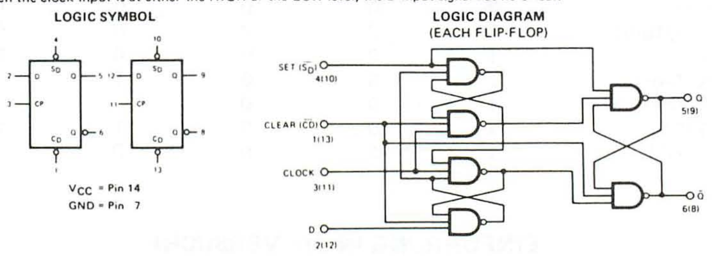
Dieser Versuch verfolgt den Zweck, eine "Bit-Auffang"-Schaltung aufzubauen, die es Ihnen ermöglicht, den Adreß- und Daten-BUS zu beobachten, während die CPU Speicherzugriffs- und Ein-/Ausgabe-Befehle ausführt.

Tabelle 3-7. Eigenschaften des T54LS74/T74LS74

DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOP

DESCRIPTION – The T54LS74/T74LS74 dual edge-triggered flip-flop utilizes Schottky TTL circuitry to produce high speed D-type flip-flops. Each flip-flop has individual clear and set inputs, and also complementary Q and $\overline{Q}$ outputs.

Information at input D is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the HIGH or the LOW level, the D input signal has no effect.



MODE SELECT – TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	$\overline{S}_D$	$\overline{C}_D$	D	Q	$\overline{Q}$
Set	L	H	X	H	L
Reset (Clear)	H	L	X	L	H
*Undetermined	L	L	X	H	H
Load "1" (Set)	H	H	h	H	L
Load "0" (Reset)	H	H	l	L	H

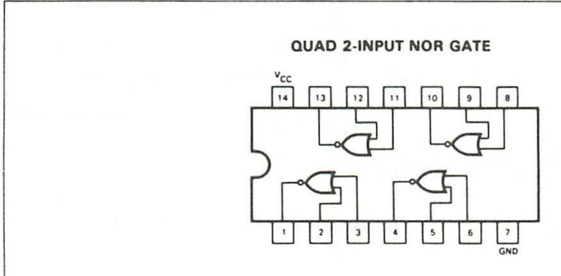
*Both outputs will be HIGH while both $\overline{S}_D$ and $\overline{C}_D$ are LOW, but the output states are unpredictable if $\overline{S}_D$ and $\overline{C}_D$ go HIGH simultaneously.

H,h = HIGH Voltage Level
L,l = LOW Voltage Level
X = Don't Care
l, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition.

Die Pinbelegung für die bei diesem Versuch verwendeten ICs ist aus den Tabellen 3-7 und 3-8 ersichtlich. Das Schaltbild für den Versuch zeigt Bild 3-26.

Tabelle 3-8. Eigenschaften des T54LS02/T74LS02

QUAD 2-INPUT NOR GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS02X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS02X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			1.6	3.2	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			2.4	5.4	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C (See Page 4-50 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	5.0	10	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	5.0	10	ns	C_L = 15 pF

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Programm LOOP1

Objekt-Code		Quell-Code	Bemerkung
D3C5	LOOP1:	NAME LOOP1 OUT (0C5H),A	;Inhalt des Akkumulators ;an Gatter C5 ausgeben
18FC		JR LOOP1	; bis zum Haltepunkt ;wiederholen oder rück- ;setzen

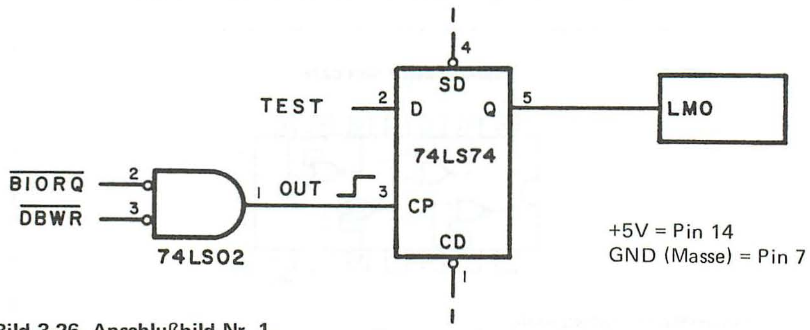


Bild 3-26. Anschlußbild Nr. 1.

1. Schritt

Verdrahten Sie die in Bild 3-26 dargestellte Schaltung. In dem Schaltbild ist das Gatter 74LS02 nicht mit dem üblichen in Bild 3-27A gezeigten NOR-Gattersymbol bezeichnet, sondern mit dem in Bild 3-27B verwandten Symbol!

Die kleinen Kreise auf den AND-Gattereingängen sind *Umkehrkreise* und werden als Äquivalent zu Invertern interpretiert. Der Ausgang des AND-Gatters befindet sich nur dann im Zustand logisch 1, wenn beide Eingänge logisch 0 sind. Dieses Verhalten entspricht genau dem des NOR-Gatters! Daher stellen die im Schaltbild gezeigten Symbole eine logische "NOR"-

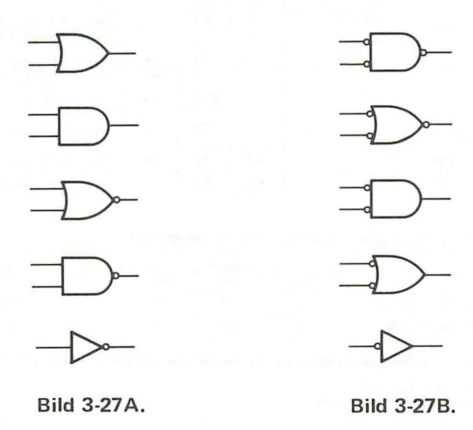


Bild 3-27A. **Bild 3-27B.**

Funktion dar. In Kapitel 4 werden alternative Gatter-Darstellungen im Detail behandelt.
Der "D"-Eingang des ICs 74LS74 ist zunächst mit der Adreß-BUS-Leitung BA7 verbunden. Dies geschieht durch Verbindung des Testdrahts mit Anschluß C, Pin 29. Er ist auf der Experimentier-Platine des Nanocomputers® mit BA7 bezeichnet. Wie bereits erwähnt, handelt es sich bei BA7 (Buffered Address Line 7 – gepufferte Adreßleitung 7) um den A7-Ausgangs-Pin der Z-80-CPU. Der Pin ist über einen Puffer mit Anschluß C verbunden.
BA7 entspricht dem Testpunkt im Schaltbild. Wenn sich der Testpunkt im Zustand logisch 1 befindet, leuchtet die LED LMO auf, sobald das Flipflop an Anschluß CP ein Taktsignal erhält; anderenfalls leuchtet die LED LMO nicht auf.

2. Schritt

Laden Sie das Programm LOOP1 in den Nanocomputer® und lassen Sie es mit normaler Geschwindigkeit ablaufen. Leuchtet dabei LED LMO auf oder bleibt sie dunkel?
Sie muß aufleuchten, weil während der Ausführung des Befehls OUT (C5H),A die Leitungen $\overline{IORQ}$ und $\overline{WR}$ der CPU aktiviert werden. Infolgedessen steht am Ausgang des Gatters 74LS02 ein positiver Impuls zur Verfügung. Es handelt sich dabei um einen OUT-Geräte-Auswahlimpuls. Er gelangt an den Takteingang CP des positiv flankengetriggerten Flipflops 74LS74, dessen Q-Ausgang dadurch auf logisch 1 geht.
Gelangt an den Takteingang des Flipflops die positive Flanke, wird das Signal am D-Eingang unverändert zum Q-Ausgang übertragen. Das heißt, die LED leuchtet auf, wenn sich der Testpunkt im Zustand logisch 1 befindet, sie bleibt dunkel, wenn sich der Testpunkt im Zustand logisch 0 befindet (siehe Bild 3-28). Sie können jetzt erkennen, warum die Schaltung *Bit-Auffänger oder Bit-Resonator* genannt wird. Die Schaltung macht den logischen Wert auf der BA7-Leitung des Adreß-BUS bei einem Ausgangszyklus deutlich. Was passiert auf dem Adreß- und Daten-BUS bei einem Ausgabezyklus?

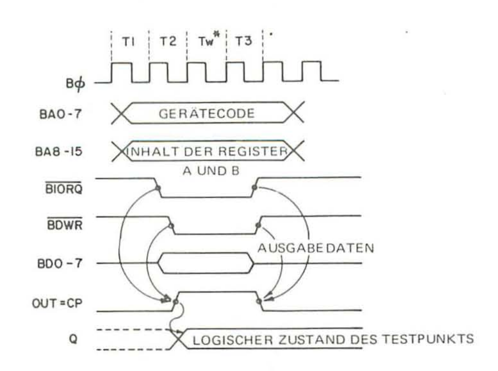


Bild 3-28. Zeitdiagramm der "Bit-Resonator-Schaltung" aus dem Bild 3-26. Es verdeutlicht den Ablauf der I/O-Schreib-Bits.

1. Der Gerätecode (in diesem Falle C5) wird auf die niederwertigere Hälfte des Adreß-BUS BA0 . . . BA7 gelegt.
2. Der Akkumulatorinhalt wird auf die höherwertige Hälfte des Adreß-BUS BA8 . . . BA15 gelegt.
3. Der Akkumulatorinhalt gelangt auf den Daten-BUS.

Da der Gerätecode C5 ist, müssen die Leitungen BA0 bis BA7 während des durch den Befehl OUT (C5H),A erzeugten Ausgangszykluses die folgenden Logikpegel aufweisen:

BA–7	6	5	4	3	2	1	0
1	1	0	0	0	1	0	1

Prüfen Sie dies, indem Sie den Testanschluß des Bit-Auffängers (Eingang D des 74LS74) mit den entsprechenden Leitungen des Adreß-BUS verbinden.

3. Schritt
Während das Programm läuft, prüfen Sie jede Daten-BUS-Leitung und halten die Ergebnisse fest:

BA–7	6	5	4	3	2	1	0

(BDn sind gepufferte Dn-Z-80-CPU-Datenleitungen!) Welchen Inhalt hat der Akkumulator?

Um Ihre Vermutung zu prüfen, drücken Sie die Taste BREAK auf dem Nanocomputer® und bringen die Auswahldiode in die AF-Stellung. (Wenn Sie die Taste RESET drücken, setzen Sie die Werte im Akkumulator und den anderen Registern auf 00 zurück, durch Drücken der BREAK-Taste werden die Register nicht berührt!) Der Inhalt des auf dem Display des Nanocomputers® angezeigten Registers A muß mit dem übereinstimmen, was der Bit-Resonator auf dem Daten-BUS dekodiert hat. Natürlich müssen Sie eine Anzeige umwandeln, entweder die Hex-Anzeige in den Binärcode oder die Binär-Anzeige in den Hexcode.

4. Schritt
Benutzen Sie nun den Bit-Auffänger; stellen Sie fest, was sich auf den höherwertigen acht Adressenleitungen befindet: (NICHT VERGESSEN, das Programm LOOP1 muß bei LOOP1 beginnen!)

BA–15	14	13	12	11	10	9	8

Auch muß hierbei der Akkumulatorinhalt sichtbar sein.

5. Schritt
Die nächsten Versuche befassen sich mit den verschiedenen Gerätecodes und Bytes im Akkumulator. Um die verschiedenen Gerätecodes zu reflektieren, geben Sie die verschiedenen Test-Bytes in den Akkumulator vor.

Füllen Sie die folgende Tabelle aus:

Geräte-code	Akku.	BA–15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FD	CC																
C1	1C																

Hierbei muß der Gerätecode mit BA0 . . . BA7 sowie der Akkumulatorinhalt mit BA8 . . . BA15 übereinstimmen.

6. Schritt

Machen Sie nun einige Bits bei einem Eingabezyklus sichtbar. Welche Veränderung müssen Sie an dem Bit-Resonator vornehmen?

Ändern Sie lediglich den Eingang zum Gatter 74LS02 von $\overline{DBWR}$ zu $\overline{BRD}$. Die neue Schaltung ist in Bild 3-29 dargestellt.

Man muß das Programm LOOP1 in LOOP2 ändern, damit es anstatt Ausgangszyklen Eingangszyklen erzeugt. Die Erklärung, weshalb der Akkumulator vor Ausführung des IN-Befehls gestartet wird, folgt zu einem späteren Zeitpunkt.

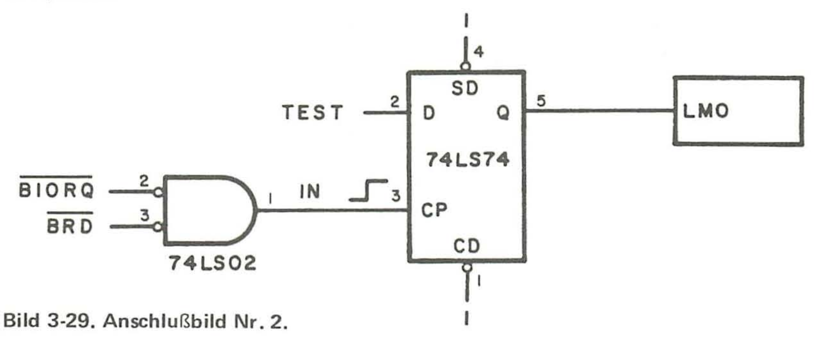


Bild 3-29. Anschlußbild Nr. 2.

Programm LOOP2

Objekt-Code		Quell-Code	Bemerkungen
		NAME LOOP2	
3E21	LOOP2:	LD A,21H	;Akkumulator starten
DBC5		IN A,(0C5H)	;ein Daten-Byte aus Gatter ;C5 eingeben
18FA		JR LOOP2	;bis zum Haltepunkt ;wiederholen oder rück- ;setzen

Was geschieht bei einem Eingabezyklus:

1. Der Gerätecode wird auf die untere (niederwertigere) Hälfte des Adreß-BUS gelegt.
2. Die obere (höherwertige) Hälfte des Adreß-BUS nimmt den Akkumulatorinhalt auf.

3. Die aktivierten $\overline{\text{IORQ}}$- und $\overline{\text{RD}}$-Signale der Z-80-CPU erzeugen einen IN-Geräte-Auswahlimpuls.
4. Die CPU tastet den Daten-BUS nach einem von einem externen Geräte als Antwort auf die Signale $\overline{\text{IORQ}}$ und $\overline{\text{RD}}$ aufgelegten Eingabe-Byte ab.

Führen Sie das obige Programm aus, indem Sie bei LOOP2 beginnen. Mit dem neu verdrahteten Bit-Auffänger können Sie die Schritte 1, 2 und 3 experimentell leicht nachprüfen. Wie Sie aus Bild 3-30 ersehen, zerlegen Sie den Daten-BUS in Abschnitte, wenn das externe Eingabegerät seine Daten auf den Daten-BUS auflegen soll, damit die CPU sie ablesen kann. Da kein Gerät mit dem Eingangsgatter C5 verbunden ist, können Sie nicht kontrollieren, was bei dem Eingangszyklus auf dem Daten-BUS ankommt. Es gibt eine Möglichkeit, den 4. Schritt zu überprüfen. Dazu muß man ein Gerät an das Eingangsgatter C5 anschließen. Dann kontrolliert man beim Eingabezyklus den Daten-BUS und überprüft anschließend den Akkumulatorinhalt, der mit dem Eingabe-Byte übereinstimmen muß. Bei diesem Versuch muß der Wert FF als Folge des Eingabefehls in den Akkumulator eingelesen werden.
Führen Sie das Programm in Einzelschritten durch und beobachten dabei das AF-Register. Damit Sie den Wechsel im Akkumulator beobachten können, startet er am Anfang jeder Schleife.

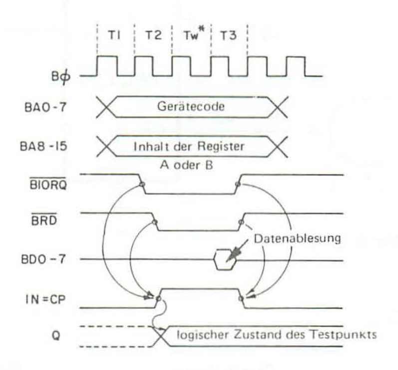


Bild 3-30. Zeitdiagramm der "Bit-Resonator-Schaltung" aus Bild 3.29. Es verdeutlicht den Ablauf der I/O-Lese-Bits.

7. Schritt

Um die Bits auch beim Speicherzugriff sichtbar zu machen, sind die Schaltungen in Bild 3-31 und 3-32 notwendig. Sie unterscheiden sich von den bisherigen Dekodierschaltungen lediglich durch die angeschlossenen Signale. Die Speicherauslesung dekodiert die Schaltung aus Bild 3-31, während für die Einlesung Bild 3-32 als Dekoder fungiert.
Erinnern Sie sich, was bei einem Speicher-Ablesezyklus geschieht:
1. Die abzulesende Adresse wird auf den Adreß-BUS gelegt.
2. Die aktivierten $\overline{\text{BMREQ}}$- und $\overline{\text{BRD}}$-Signale erzeugen ein MEMR-Adressensignal.

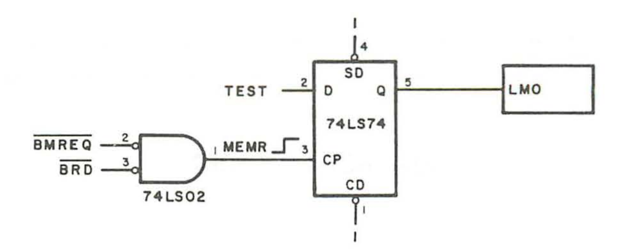


Bild 3-31. Anschlußbild Nr. 3. Dekoder für die Speicherauslesung.

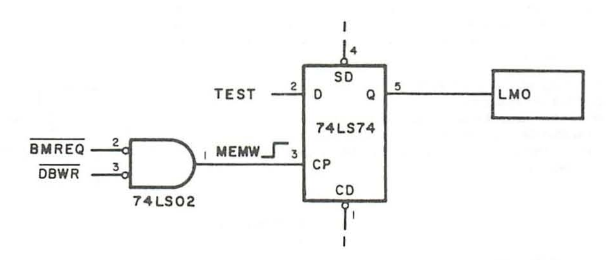


Bild 3-32. Anschlußbild Nr. 4. Dekoder für die Speicherauslesung.

3. Die adressierte Speicherstelle legt ihren Inhalt auf den Daten-BUS.
4. Dei CPU liest den Inhalt der Speicherstelle vom Daten-BUS ab.

Es besteht zweifellos eine große Ähnlichkeit zwischen den Funktionen Speicherzugriff und Geräte-I/O. Es gibt jedoch einen Unterschied, der die Bit-Auffang-Schaltung wesentlich beeinflußt. Der Bit-Auffänger für die I/O-Bytes wird bei jedem Ein- oder Ausgabezyklus aktiviert.
Ähnlich zeigt Anschlußbild Nr. 3 in Bild 3-31 eine Schaltung, die bei *jedem* Speicher-Lesezyklus aktiv wird. Die in Bild 3-32 verwendeten Signale aktivieren die Schaltung bei *jedem* Speicher-Schreibzyklus. Es ist somit deutlich, warum die Bit-Auffang-Schaltung aus Bild 3-31 nicht zum Auffangen von Adressen und Eingabe-Bytes für Speicher-Lesebefehle wie LD A,(HL) geeignet ist: Bei der Programmausführung beginnt *jeder* Befehl mit einem M1- oder Befehl-Abrufzyklus. Ein $\overline{M1}$-Zyklus ist nichts anderes als ein Speicher-Lesezyklus mit aktiviertem M1-Signal. Deshalb beinhaltet jeder Befehl bei einer Programmausführung mindestens einen Speicher-Lesezyklus, der den Bit-Auffänger aktiviert. Bei Ausführung einer unendlichen Schleife ist der Bit-Auffänger daher nicht in der Lage, irgendein Adreß- oder Daten-Bit zu blockieren, da sie sich ständig ändern.
Wäre es dann sinnvoll, das Anschlußbild Nr. 3 (Bild 3-31) um eine Logikschaltung zu erweitern, die das Triggern des Bit-Auffängers bei M1-Zyklen verhindert?
Die Antwort lautet NEIN. Der M1-Zyklus findet nur bei Objektcode-Abrufen statt. Viele Befehle enthalten außer Objektcodes noch andere Bytes (z.B. LD A, < B2 >). Diese "Nicht-Objektcode"-Bytes werden mit Speicher-Lesezyklen abgerufen, wobei das M1-Signal nicht aktiv ist.

Tabelle 3-9. Eigenschaften des T54LS139/T74LS139

DUAL 1-OF-4 DECODER

DESCRIPTION – The LSTTL/MSI T54LS139/T74LS139 is a high speed Dual 1-of-4 Decoder/Demultiplexer. The device has two independent decoders, each accepting two inputs and providing four mutually exclusive active LOW outputs. Each decoder has an active LOW Enable input which can be used as a data input for a 4-output demultiplexer. Each half of the LS139 can be used as a function generator providing all four minterms of two variables. The LS139 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- SCHOTTKY PROCESS FOR HIGH SPEED
- MULTIFUNCTION CAPABILITY
- TWO COMPLETELY INDEPENDENT 1-OF-4 DECODERS
- ACTIVE LOW MUTUALLY EXCLUSIVE OUTPUTS
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

PIN NAMES		LOADING (Note a)	
		HIGH	LOW
A_0, A_1	Address Inputs	0.5 U.L.	0.25 U.L.
$\bar{E}$	Enable (Active LOW) Input	0.5 U.L.	0.25 U.L.
$\bar{O}_0 - \bar{O}_3$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM

V_{CC} = Pin 16
GND = Pin 8
O = Pin Numbers

LOGIC SYMBOL

V_{CC} = Pin 16
GND = Pin 8

CONNECTION DIAGRAM
DIP (TOP VIEW)

Die Folgerung: Der vorliegende Bit-Auffänger ist zumindest zum Auffangen von Speicher-Lesedaten und Adressen ungeeignet. Das Grundproblem ist also das unselektive Triggern der Schaltung. Zur Lösung dieses Problems sind verschiedene Möglichkeiten denkbar. Eine weitere Untersuchung des Bit-Auffängers ist Ihnen selbst überlassen. Vielleicht möchten Sie z.B. auch Speicher-Schreibdaten und Adressen auffangen.

FUNCTIONAL DESCRIPTION – The LS139 is a high speed dual 1-of-4 decoder/demultiplexer fabricated with the Schottky barrier diode process. The device has two independent decoders, each of which accept two binary weighted inputs (A_0, A_1) and provide four mutually exclusive active LOW outputs ($\bar{O}_0$-$\bar{O}_3$). Each decoder has an active LOW Enable ($\bar{E}$). When $\bar{E}$ is HIGH all outputs are forced HIGH. The enable can be used as the data input for a 4-output demultiplexer application.

Each half of the LS139 generates all four minterms of two variables. These four minterms are useful in some applications, replacing multiple gate functions as shown in Fig. a, and thereby reducing the number of packages required in a logic network.

TRUTH TABLE

INPUTS			OUTPUTS			
$\bar{E}$	A_0	A_1	$\bar{O}_0$	$\bar{O}_1$	$\bar{O}_2$	$\bar{O}_3$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

Fig. a

VERSUCH NR. 2

Dieser Versuch hat den Zweck, eine Schaltung zu bauen, die vier Eingabe-Gerätecodes und vier Ausgabe-Gerätecodes dekodiert. Zur Prüfung dieser Schaltung können Sie zum Teil den Bit-Auffänger aus Versuch Nr. 1 verwenden.
Die zum Versuch notwendige Schaltung zeigt Bild 3-33; die Pinbelegung für das IC 74LS139 zeigt die Tabelle 3-9.

Programm DECODE

Objekt-Code		Quell-Code	Bemerkungen
		NAME DECODE	
0E20	DECODE:	LD C,20H	;Gerätecode in Register C ;eingeben
06C5		LD B,C5H	;ein geeignetes Byte in ;Register B eingeben – ;zur anschließenden Be- ;obachtung auf der ;oberen Hälfte des Adreß- ;BUS
ED61	LOOP3:	OUT (C),H	;Inhalt des H-Registers ;über das von Register C ;bezeichnete Gatter aus- ;geben
18FC		JR LOOP3	;Ausgabebefehl bis zum ;Haltepunkt wiederholen ;oder rücksetzen

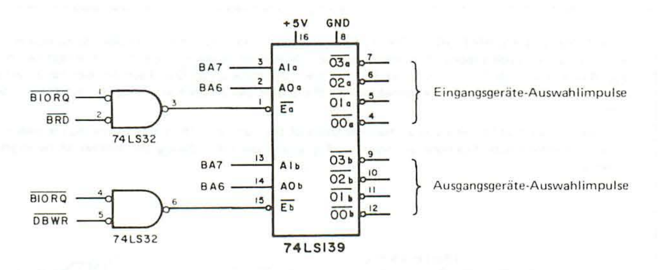


Bild 3-33. Dekoderschaltung für vier Eingabe- und Ausgabe-Gerätecodes.

1. Schritt

Bauen Sie die im Schaltbild (Bild 3-33) dargestellte Schaltung auf. Falls Sie nicht über den 74LS74-D-Flipflop-"Bit-Auffänger" ZUR BIT-BLOCKIERUNG BEIM AUSGABEZYKLUS verfügen, bauen Sie auch diese Schaltung auf (siehe Versuch Nr. 1, Bild 3-26).

2. Schritt

Geben Sie das Programm ein und führen es aus. Benutzen Sie den Test-Anschluß in der Bit-Auffang-Schaltung, um alle Bits auf dem Adreß-BUS zu testen. Tragen Sie die Ergebnisse in die Tabelle ein:

BA–15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Es muß sich folgendes Ergebnis zeigen:

1. Der Gerätecode 20 steht auf den Leitungen BA0 . . . BA7 zur Verfügung.
2. Der Inhalt des B-Registers C5 erscheint auf BA8 . . .BA15:

BA–15	14	13	12	11	10	9	8
1	1	0	0	0	1	0	1

3. Schritt

Prüfen Sie jedes Bit auf dem Daten-BUS und tragen anschließend das Ergebnis ein:

BD–7	6	5	4	3	2	1	0

Sie sind jetzt in der Lage, den Inhalt des H-Registers anzugeben. Halten Sie das Programm an, setzen Sie die Auswahldiode auf HL und prüfen, ob der Inhalt des H-Registers mit den auf dem Daten-BUS festgestellten Daten übereinstimmt. (ACHTUNG: Benutzen Sie die Break-anstelle der RESET-Taste, da sich sonst der Registerinhalt ändert!) Starten Sie das Programm erneut. Sie müßten auf dem Daten-BUS 21H feststellen.

4. Schritt

Die Freigabe-Eingänge $\overline{Ea}$ (Pin 1) und $\overline{Eb}$ (Pin 15) des Dekoder/Demultiplexers 74LS139 von den Gatterausgängen 74LS32 abtrennen und mit Masse verbinden. Dadurch sind beide Dekoder ständig freigegeben. Starten Sie das Programm erneut. Das Ziel ist jetzt, die logischen Zustände des 74LS139 während der Ausführung des Befehls OUT (C),H festzustellen. Idealerweise ließe sich dies am einfachsten bewerkstelligen, indem man den Testanschluß der Bit-Auffang-Schaltung mit den Ausgangspins des 74LS139 verbindet. Die folgende Analyse macht deutlich, ob diese Annahme stimmt. Vergewissern Sie sich zunächst, ob der Bit-Auffänger den richtigen Befehl OUT (C),H befolgt. Beide Dekoder/Demultiplexer im 74LS139 sind ständig aktiv, daher können sie kaum selektiv werden. Was ist mit dem Bit-Auffänger selbst? Er ist konzipiert, um Bits ''aufzufangen'', wenn $\overline{DBWR}$ und $\overline{IORQ}$ gleichzeitig aktiv sind. Dies ist aber nur bei I/O-Schreibzyklen der Fall. Gibt es in dem Programm DECODE Befehle, welche die Z-80-CPU zu I/O-Schreibzyklen veranlaßt? Die Antwort ist ja; es ist der Befehl OUT (C),H. Hieraus können Sie schließen, der Bit-Auffänger ''fängt'' nur dann Bits auf, während dieser Befehl gerade ausgeführt wird.
Was passiert also? Welche logischen Zustände nehmen die Ausgänge an? Der 74LS139 hat zwei Sätze von vier Ausgängen, die normalerweise high sind. In Abhängigkeit von den logischen Werten auf den beiden Auswahl-Eingängen A0a (Pin 2) und A1a (Pin 3) für den Dekoder/Demultiplexer ''a'' und A0b (Pin 14) sowie A1b Pin 13) für den Dekoder/Demultiplexer ''b'' wird ein Pin pro Satz bei einem durch den OUT (C),H-Befehl erzeugten I/O-Schreibzyklus low. Aber welcher? Da Sie den Inhalt des Adreß-BUS kennen, wissen Sie auch die Antwort. BA0 . . . BA7 enthält den Gerätecode von Register C, der 20 lautet. Darum befinden sich sowohl BA6 als auch BA7 im Zustand logisch 0, während der Befehl OUT (C),H ausgeführt wird. Dies läßt auf einen Low-Zustand von 00a (Pin 4) und 00b (Pin 12) schließen. Die anderen 74LS139-Ausgänge 5, 6, 7, 9, 10 und 11 sind high.

Was die Dekoder-Ausgänge betrifft, kann man also folgendes erwarten:

Pin 4: logisch 0	Pin 12: logisch 0
Pin 5: logisch 1	Pin 11: logisch 1
Pin 6: logisch 1	Pin 10: logisch 1
Pin 7: logisch 1	Pin 9: logisch 1

Verwenden Sie den Bit-Auffänger (für die Ausgangszyklen), um diese Voraussage zu prüfen. Ihre Feststellungen müßten die Theorie bestätigen.

5. Schritt

Testen Sie die folgenden Gerätecodes, um die Auswirkung auf die Ausgänge des Dekoder-ICs zu bestimmen:

40 60 80 C0

Bei den Gerätecodes 40 und 60 sind die Anschlußpins 5 und 11 logisch 0; der Gerätecode 80 dekodiert bei den Anschlußpins 6 und 10 eine logische

0 und bei dem Code C0 ist der Logikpegel an den Pins 7 und 9 gleich "0". In allen Fällen ist bei jedem Dekoder ein Pin logisch 0, während alle anderen auf logisch 1 sind. An dieser Stelle wird deutlich: Diese Dekoderschaltung dekodiert die niedrigwertigsten acht Bits des Adreß-BUS *nicht* absolut:

a) Nur zwei Leitungen, A6 und A7 sind dekodiert.
b) Zwei unterschiedliche Gerätecodes (40/60) dekodieren denselben Auswahlimpuls.

6. Schritt

Ändern Sie den Bit-Auffänger für Eingabezyklen gemäß Bild 3-29, Anschlußbild Nr. 2 und den Befehl OUT (C),H im Programm zu IN H,(C), Hex-Code ED 60. Jetzt prüfen Sie, ob die Beobachtungen, die Sie für I/O-Schreibzyklen gemacht haben, auch für I/O-Lesezyklen zutreffen. Zu diesem Zwecke ändern Sie den Gerätecode wieder wie zuvor und stellen die logischen Zustände an den Ausgängen des Dekoder/Demultiplexers fest. Bei der Wahl der Testgerätecodes ist bewußt der Bereich 00 . . . 1F ausgespart; dieser Bereich ist für den Nanocomputer® reserviert.

7. Schritt

Verbinden Sie die Freigabe-Eingänge $\overline{Ea}$ (Pin 1) und $\overline{Eb}$ (Pin 15) des Dekoder/Demultiplexers 74LS139 mit den beiden 74LS32-OR-Gattern, wie die Schaltung in Bild 3-33 zeigt. Jetzt ist der Dekoder/Demultiplexer "a" bei I/O-Lese-(oder IN-)operationen aktiv, während der Dekoder/Demultiplexer "b" nur bei I/O-Schreib-(oder OUT-)operationen aktiv ist. Dies ist eine nützliche Dekoder-Schaltung. Es stellt sich die Frage, warum die Schaltung erst jetzt eingesetzt wird. Der Grund ist die *Zeitfolge*. Bei den vorhergehenden Schritten dient eine Bit-Auffang-Schaltung zur Beobachtung der Ausgänge des 74LS139. Für die in Bild 3-33 gezeigte Schaltung ist der Bit-Auffänger zu schnell, um die logischen Ausgangszustände bei I/O-Lese- oder-Schreibzyklen aufzufangen. Der Bit-Auffänger "fangt" den logischen Zustand an den 74LS139-Ausgängen, *bevor* sie eine Chance haben, den Zustand der Eingänge richtig zu reflektieren. Nachstehend eine Auflistung aller Ereignisse die eintreten, wenn Sie den logischen Zustand der 74LS139-Ausgänge (Bild 3-33) mit dem Bit-Auffänger messen würden. Bild 3-34 ist ein Zeitdiagramm, aus dem die relativen Ereignisse für einen I/O-Schreibzyklus nach der Zeitfolge ersichtlich sind.

1. Der IN (C), H oder OUT (C),H Befehl wird von der CPU ausgeführt, so gilt bei z.B. dem OUT (C), H Befehl:
   a) Der Gerätecode wird auf BA0 . . . BA7; der Inhalt des B-Registers auf BA8 . . . BA15 gelegt.
   b) Die Signale $\overline{BIORQ}$ und $\overline{DBWR}$ werden aktiviert.
2. Die Signale $\overline{BIORQ}$ und $\overline{DBWR}$ dienen als Eingangsimpuls für:
   74LS02: NOR für die Bit-Auffang-Schaltung
   74LS32: OR für die Dekoder-Schaltung

   Jedes dieser Gatter erzeugt ein Ausgangssignal, um ein D-Flipflop oder einen Dekoder/Demultiplexer zu takten oder freizugeben. Die Erzeugung dieses Ausgangsimpules nimmt *Zeit* in Anspruch.
   Dieser Zeitabschnitt wird *Verzögerungszeit* genannt und ist in Bild 3-24

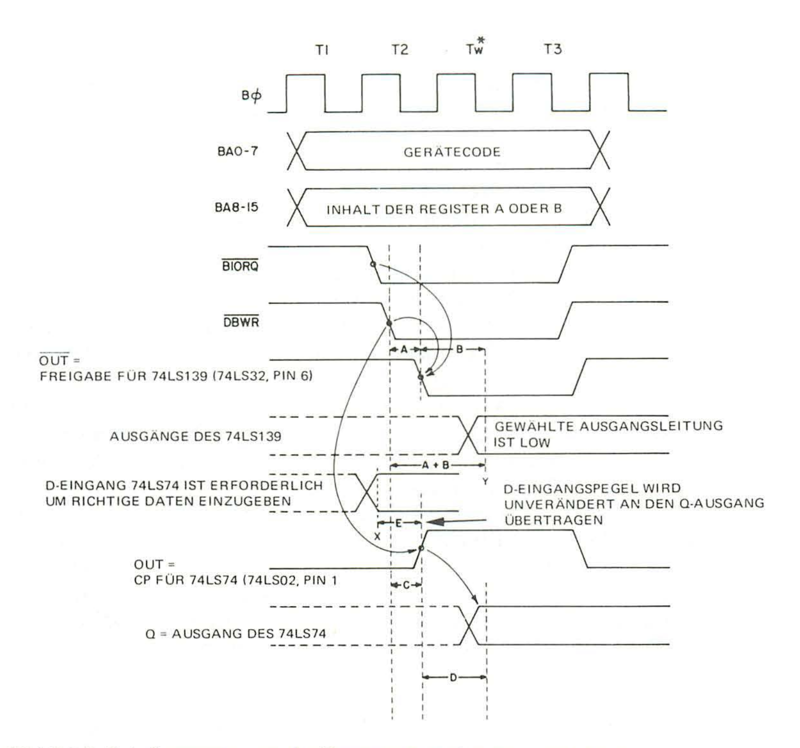


Bild 3-34. Zeitdiagramm zum Auffangen von Dekoder-Ausgangs-Bits.

für das 74LS32-OR-Gatter mit A und für das 74LS02-NOR-Gatter mit C bezeichnet. In den Tabellen 3-8 und 3-9 sind die maximalen (MAX), minimalen (MIN) und typischen Verzögerungszeiten für jedes IC angegeben. Bei Übergängen von low auf high oder umgekehrt sind die Verzögerungen unterschiedlich. Beim 74LS02 ist der Ausgang normalerweise low und der interessierende Übergang die positive Flanke. Das Datenbuch macht für die Verzögerungszeiten folgende Angaben:

$$3{,}0 \text{ ns} < C < 10{,}0 \text{ ns und typisch} = 5{,}0 \text{ ns}$$

Beim 74LS32 ist der Ausgang im Ruhezustand high und der interessierende Übergang von high nach low, also die negative Flanke. Es gilt:

$$3{,}0 \text{ ns} < A < 11{,}0 \text{ ns und typisch} = 7{,}0 \text{ ns}$$

Es macht sich bereits ein potentieller Unterschied darin bemerkbar, wie schnell die Signale die Dekoder- und Bit-Auffang-Schaltung durchlaufen.

3. Der Ausgang des 74LS32 gibt den Dekoder/Demultiplexer "b" des 74LS139 frei. Auf Grund der Verzögerungszeit gibt es vor den Ausgängen $\overline{00b}$, $\overline{01b}$, $\overline{02b}$ und $\overline{03b}$, die den Inhalt des Auswahl-Eingangs

A0b und A1b dekodieren, eine Verzögerung. Der Adreß-BUS hat den Gerätecode stabilisiert, lange bevor der Freigabeimpuls den 74LS139-Chip erreicht! In Bild 3-34 ist die Verzögerungszeit mit B bezeichnet. Auch hier ist der Übergang von high auf low von primären Interesse. Aus einem Datenbuch: B < 24 ns und typisch = 17 ns (keine MIN-Angabe) Folglich A+B = Verzögerung in ns durch die Aktivierung von
   $\overline{\text{BIORQ}}$ und $\overline{\text{DBWR}}$ zur Stabilisation des 74LS139-Ausgangs.
   A+B < 35 ns und typisch = 24 ns

4. Der Ausgangsimpuls des 74LS02 taktet das 74LS74-Flipflop. Der D-Eingangsimpuls (Pin 2) wird unverändert an den Ausgang Q mit der in Bild 3-34 als D bezeichneten Verzögerungszeit übernommen. Diese Verzögerung ist in der Tat unkritisch. Kritisch ist, ob die Dekoder/Demultiplexer-Ausgänge sich stabilisiert haben oder nicht. Das heißt: Der Zustand des Auswahl-Eingangs muß dekodiert sein, bevor der 74LS74 seinen D-Eingangspegel unverändert zum Q-Ausgang überträgt. Leider haben sie sich nicht notwendigerweise stabilisiert. Was noch schlimmer ist, der D-Eingang des Flipflops muß die Daten mindestens 20 ns vor dem Impuls zum CP-Eingang erhalten. Das D-Flipflop beginnt den augenblicklichen D-Eingangspegel fast zur gleichen Zeit zum Q-Ausgang zu übertragen, in welcher der Dekoder/Demultiplexer abgetastet wird (innerhalb von 0 . . . 8 ns). Aus Bild 3-34 ist leicht zu erkennen, wann der Übertrag erfolgt, nämlich möglicherweise lange bevor bei Punkt Y der gewählte Ausgang des 74LS139 auf low geht (X lange vor Y).

Somit gibt es noch eine weitere Begrenzung in der Anwendung der Bit-Auffang-Schaltung. Wie die meisten einfachen, billigen Hilfsmittel hat sie ihre Vorteile, ist aber nicht universell einsetzbar. Besonders, wenn Signale hintereinander durch eine Reihe von Geräten geschickt werden, können Zeitprobleme recht kompliziert sein. Wie Sie sich vorstellen können, sind Zeitanalysen für komplizierte Schaltungen oft recht komplex. Denken Sie nur an die Zeit-Anforderungen für den Z-80! In der Entwicklungsphase werden bei der Zeitfolgeeinstellung für solche Schaltungen vielfach Computer eingesetzt.

Testen Sie Ihren Bit-Auffänger an den 74LS139-Ausgängen sowohl für die I/O-Lese- als auch die -Schreiboperation. Stellen Sie ein fehlerhaftes Verhalten fest? Wenn nicht, sind die Verzögerungszeiten so günstig, daß alles einwandfrei funktioniert.
Für den Versuch Nr. 3 ist die Schaltung aus Bild 3-33 erforderlich.

VERSUCH NR. 3

Dieser Versuch macht die Anwendung von Geräte-Auswahlimpulsen deutlich, die von der Software durch Abtasten eines Dekadenzählers gesteuert werden. Der Dekadenzähler 74LS90 zählt die Mehrfach-Geräte-Auswahlimpulse, für deren Erzeugung man auch den OTIR-Befehl benutzt.

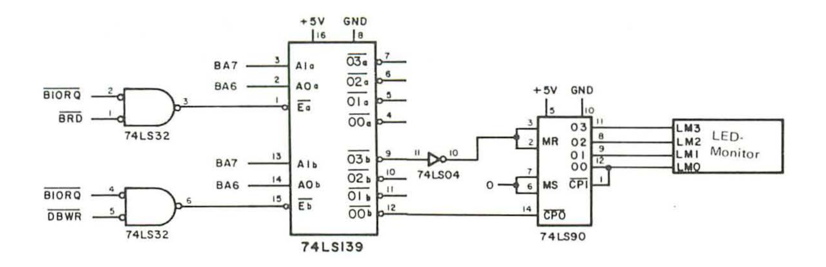


Bild 3-35. Geräte-Auswahlimpuls-Schaltung mit dem Dekadenzähler 74LS90.

Programm PULSR

Object-Code		Quell-Code	Bemerkungen
		NAME PULSR	
0E20	PULSR:	LD C,20H	;Gerätecode in Register C ;eingeben
21000F		LD HL, TABLE	;startende Speicheradresse ;in Registerpaar HL ein- ;geben
0608		LD B,08H	;Byte-Zähler in Register ;B eingeben
D3C0		OUT (0C0H), A	;Dekadenzähler auf 0 ;setzen
EDB3		OTIR	;Byte-Folge ausgeben, ;beginnend bei Adresse ;HL der Länge (B) zum ;zum Gatter (C)
76		HALT	;CPU anhalten

1. Schritt

Schaltung gemäß Bild 3-35 verdrahten. Falls Sie die Dekoder-Schaltung aus Versuch 2 nicht aufgebaut haben, holen Sie dies nach. Schließen Sie Pin 12 des Dekoders 74LS139 an Pin 14 des Dekadenzählers 74LS90 und Pin 9 des 74LS139 über einen Inverter an Pin 2 und 3 des 74LS90 an. Die Anschlüsse 6 und 7 des 74LS90 verbinden Sie mit 0 (Masse).

2. Schritt

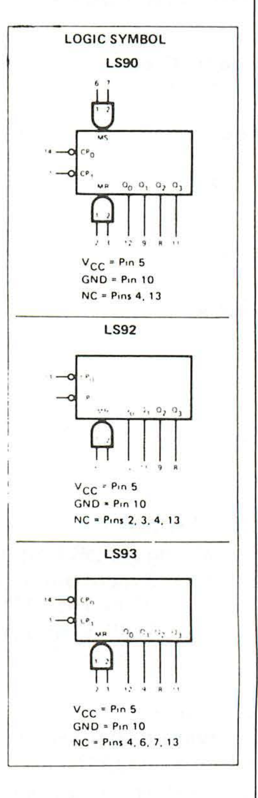
Programm PULSR in den Nanocomputer® eingeben und ausführen. Was beobachten Sie?
Der Zähler müßte acht registriert haben, was der vorm OTIR-Befehl ausgegebenen Byte-Zahl und somit der Anzahl erzeugter Geräte-Auswahlimpulse entspricht.

3. Schritt

Ändern Sie den Befehl LD B,08H, damit das B-Register alle folgenden

Tabelle 3-10. Eigenschaften der Dekadenzähler T54LS90/T74LS90, T54LS92/T74LS92 und T54LS93/T74LS93.

DESCRIPTION The T54LS90/T74LS90, T54LS92/T74LS92 and T54LS93/T74LS93 are high-speed 4-bit ripple type counters partitioned into two sections. Each counter has a divide-by-two section and either a divide-by-five (LS90), divide-by-six (LS92) or divide-by-eight (LS93) section which are triggered by a HIGH-to-LOW transition on the clock inputs. Each section can be used separately or tied together (Q to $\overline{CP}$) to form BCD, bi-quinary, modulo-12, or modulo-16 counters. All of the counters have a 2-input gated Master Reset (Clear), and the LS90 also has a 2-input gated Master Set (Preset 9).



- LOW POWER CONSUMPTION . . . TYPICALLY 45 mW
- HIGH COUNT RATES . . . TYPICALLY 50 MHz
- CHOICE OF COUNTING MODES . . . BCD, BI-QUINARY, DIVIDE-BY-TWELVE, BINARY
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

PIN NAMES		LOADING (Note a)	
		HIGH	LOW
$\overline{CP}_0$	Clock (Active LOW going edge) Input to ÷2 Section	3.0 U.L.	1.5 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷5 Section (LS90), ÷6 Section (LS92)	2.0 U.L.	2.0 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷8 Section (LS93)	1.0 U.L.	1.0 U.L.
MR_1, MR_2	Master Reset (Clear) Inputs	0.5 U.L.	0.25 U.L.
MS_1, MS_2	Master Set (Preset-9, LS90) Inputs	0.5 U.L.	0.25 U.L.
Q_0	Output from ÷2 Section (Notes b & c)	10 U.L.	5(2.5) U.L.
Q_1, Q_2, Q_3	Outputs from ÷5 (LS90), ÷6 (LS92), ÷8 (LS93) Sections (Note b)	10 U.L.	5(2.5) U.L.

NOTES

a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.
c. The Q_0 Outputs are guaranteed to drive the full fan-out plus the CP_1 input of the device.

Tabelle 3-10

FUNCTIONAL DESCRIPTION – The LS90, LS92, and LS93 are 4-bit ripple type Decade, Divide-By-Twelve, and Binary Counters respectively. Each device consists of four master/slave flip-flops which are internally connected to provide a divide-by-two section and a divide-by-five (LS90), divide-by-six (LS92), or divide-by-eight (LS93) section. Each section has a separate clock input which initiates state changes of the counter on the HIGH-to-LOW clock transition. State changes of the Q outputs do not occur simultaneously because of internal ripple delays. Therefore, decoded output signals are subject to decoding spikes and should not be used for clocks or strobes. The Q_0 output of each device is designed and specified to drive the rated fan-out plus the $\overline{CP_1}$ input of the device.

A gated AND asynchronous Master Reset ($MR_1 \bullet MR_2$) is provided on all counters which overrides and clocks and resets (clears) all the flip-flops. A gated AND asynchronous Master Set ($MS_1 \bullet MS_2$) is provided on the LS90 which overrides the clocks and the MR inputs and sets the outputs to nine (HLLH).

Since the output from the divide-by-two section is not internally connected to the succeeding stages, the devices may be operated in various counting modes.:

LS90

A. BCD Decade (8421) Counter – The $\overline{CP_1}$ input must be externally connected to the Q_0 output. The $\overline{CP_0}$ input receives the incoming count and a BCD count sequence is produced.

B. Symmetrical Bi-quinary Divide-By-Ten Counter – The Q_3 output must be externally connected to the $\overline{CP_0}$ input. The input count is then applied to the $\overline{CP_1}$ input and a divide-by-ten square wave is obtained at output Q_0.

C. Divide-By-Two and Divide-By-Five Counter – No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function ($\overline{CP_0}$ as the input and Q_0 as the output). The $\overline{CP_1}$ input is used to obtain binary divide-by-five operation at the Q_3 output.

LS92

A. Modulo 12, Divide-By-Twelve Counter – The $\overline{CP_1}$ input must be externally connected to the Q_0 output. The $\overline{CP_0}$ input receives the incoming count and Q_3 produces a symmetrical divide-by-twelve square wave output.

B. Divide-By-Two and Divide-By-Six Counter – No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function. The $\overline{CP_1}$ input is used to obtain divide-by-three operation at the Q_1 and Q_2 outputs and divide-by-six operation at the Q_3 output.

LS93

A. 4-Bit Ripple Counter – The output Q_0 must be externally connected to input $\overline{CP_1}$. The input count pulses are applied to input $\overline{CP_0}$. Simultaneous divisions of 2, 4, 8, and 16 are performed at the Q_0, Q_1, Q_2, and Q_3 outputs as shown in the truth table.

B. 3-Bit Ripple Counter – The input count pulses are applied to input $\overline{CP_1}$. Simultaneous frequency divisions of 2, 4, and 8 are available at the Q_1, Q_2, and Q_3 outputs. Independent use of the first flip-flop is available if the reset function coincides with reset of the 3-bit ripple-through counter.

Ausgabe-Bytes zählt; Programm für jeden neuen Zählvorgang ausführen; Ergebnis durch Ablesen der Leuchtdioden am 74LS90 festhalten.

Inhalt des B-Registers	Zählerwert (dezimal)
01	
02	
03	
04	
05	
06	
07	
08	
09	
0A	
0B	
0C	
0D	
0E	
0F	
10	
11	
12	
13	
14	

Für die Hexzahlen 01 . . . 09 dekodiert der Dekadenzähler die Ziffern 1 . . . 9 im Binärcode. Lädt man die Hexzahl 0A in das B-Register, liest der Zähler 0; es leuchtet die LED LM0. Für 0BH liest der Zähler 1, für 0CH = 2, für 0DH = 3 usw. Ist der Inhalt des B-Registers 13H, dekodiert der 74LS90 die Ziffer 9; es leuchten die LEDs LM1 und LM3. Gibt man in das B-Register 14H ein, springt der Zähler wieder auf Null zurück. Das heißt: Der Zähler 74LS90 zählt immer eine Dekade von 0 . . . 9 durch und beginnt mit dem nächsten Zählimpuls wieder bei 0. Man nennt ihn deshalb auch *Dekadenzähler* oder *Teiler-durch-10.*
Wie verhält sich der B-Registerinhalt zum Inhalt des Dekadenzähler? Das B-Register legt die Zahl der vom OTIR-Befehl auszuführenden Ausgangszyklen fest. Jeder Ausgangszyklus verursacht an Pin $\overline{00b}$ des Dekoders 74LS139 einen Geräte-Auswahlimpuls. Dieser ist mit dem Takteingang $\overline{CP0}$ des Zählers 74LS90 verbunden. Dadurch wird jeder Ausgangsimpuls gezählt und von den LED-Monitoren als ansteigende Binärzahl bis max. 1001 angezeigt.
Laden Sie ins Register B 20H und führen das Programm mit der SS-Taste durch. Stellen Sie dabei die Auswahl-LED auf BC und vergleichen den Inhalt des B-Registers mit der Anzeige der LEDs LM0 . . . LM3. Sie stellen fest, daß wertmäßig beide Anzeigen übereinstimmen; nur einmal im Hex- und einmal im Binärcode.

4. Schritt

Welche Funktion hat die Verbindung von Pin 9 des Dekoders zu den Pins 2 und 3 des Zählers?

Der Befehl OUT (0C0H),A verursacht an Pin 9 des Dekoders einen negativen Impuls. Der Inverter 74LS04 macht daraus einen positiven Impuls, der an die Clear- oder Reset-Eingänge des Zählers gelangt und ihn auf 0000 stellt. Die Verbindung setzt den Zähler also mit Hilfe der Software auf 0 zurück, bevor der 74LS90 mit dem Zählen der vom OTIR-Befehl erzeugten Ausgangsimpulse beginnen kann.

5. Schritt
Durch Verändern des in Register C eingegebenen Gerätecodes können Sie die Übereinstimmung des C-Registerinhaltes mit dem Dekoder/Demultiplexer 74LS139 erzeugten Geräte-Auswahlimpuls feststellen. Damit der Geräte-Auswahlimpuls bei dem neuen Gerätecode zum Takteingang des Zählers gelangt, muß man ebenfalls die Verdrahtung entsprechend ändern.

6. Schritt
Führen Sie den Versuch analog mit dem INDR-Befehl und dem Dekoder "a" des 74LS139 aus. Ähnliche Versuche können auch mit den Befehlen OTDR und INIR durchgeführt werden.

7. Schritt
Tauschen Sie den HALT-Befehl im obigen Programm gegen den Restart-Befehl RST 38H (hex. FF) aus. Führen Sie das veränderte Programm mit verschiedenen ins B-Register eingespeicherten Bytes aus, d.h., veränderen Sie das Byte in dem Befehl LD B,08H. Was beobachten Sie?

Der Ausgang des Dekadenzählers 74LS90 zeigt stets 0 an. Warum verursacht eine solch einfache Veränderung ein derartiges Ergebnis? Sie haben über den Befehl RST 38H die Steuerung des Nanocomputers® an sein Betriebssystem zurückgegeben. In diesem Falle, wenn dem Nanocomputer®-Betriebssystem die Steuerung zurückgegeben wird, beginnt er sofort seine Tastenfeld-Anzeige auf den letzten Stand zu bringen und die Tastatur nach gedrückten Tasten abzususchen. Zu diesen Aufgaben gehört auch die Geräte-I/O, d.h. wiederholte Ausführungen von IN- und OUT-Befehlen. Da diese Schaltung die Gerätcodes nicht *absolut* dekodiert, hat sie die Tendenz, öfter aktiviert zu werden als nötig. Das gilt insbesondere für den Reset-Impuls. Er steht zur Verfügung, wenn sich BA7 und BA6 beide im Zustand logisch 0 befinden. Das ist bei jedem IN- und OUT-Befehl, den das Betriebssystem des Nanocomputers® bei der Wahrnehmung seiner Tastenfeld- und Anzeige-I/O-Aufgaben durchführt, der Fall. Falls der OTIR-Befehl den Zähler zu einer Nichtnull-Zählung veranlaßt, dauert dies nur einige Millisekunden, weil das Betriebssystem durch den Befehl RST 38H die Steuerung übernimmt und den Zähler auf Null stellt. Erkennen können Sie nur die Wirkung des letzteren.
Die Beschreibung des obigen Phänomens soll Ihnen zeigen, welche Fehler entstehen können, wenn man Software verkehrt einsetzt. Sie sollte nur dann benutzt werden, wenn man völlig mit ihr vertraut ist. Leider ist dies fast nie der Fall. Deshalb ist immer äußerste Vorsicht geboten.

VERSUCH NR. 4

Dieser Versuch macht Sie mit dem Gebrauch eines 74LS30-NAND-Gatters mit acht Eingängen vertraut. Dadurch haben Sie wahlfreien Zugriff zu 256 Bytes statischer Lese/Schreibspeicher (R/W RAM).
Das R/W RAM besteht aus zwei statischen Intel-2101A-R/W RAM mit 4 x 256 Bits. Für diesen Versuch ist die Verdrahtung einer relativ komplizierten Schaltung notwendig, aber es lohnt die Mühe. Beachten Sie die Hinweise und lassen Sie sich beim Aufbau genügend Zeit.

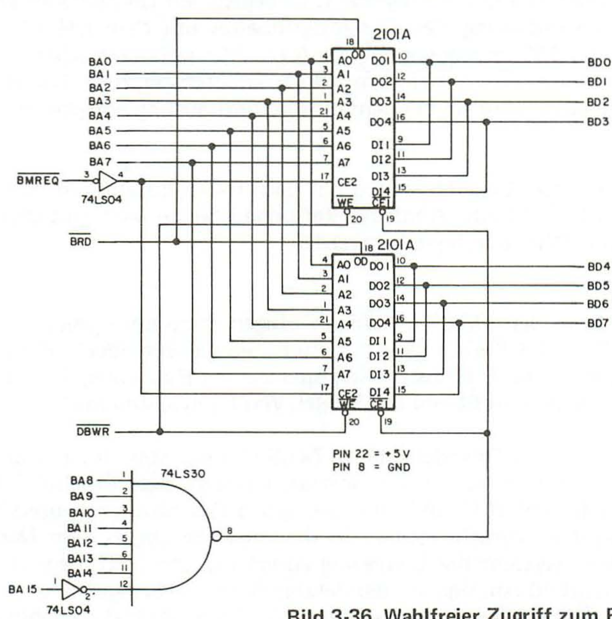


Bild 3-36. Wahlfreier Zugriff zum R/W-RAM.

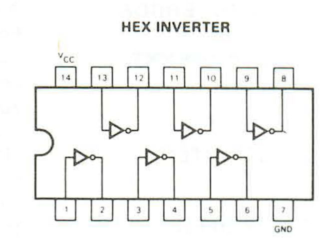
Programm MEM1

Objekt-Code		Quell-Code	Bemerkungen
		NAME MEM1	
3EFF	MEM1:	LD A,0FFH	;Akkumulator starten
3C	LOOP4:	INC A	;Speichertest für ;nächsten Wert beginnen
32007F		LD (7F00H),A	;Speicherstelle 7F00 ;zum Inhalt von A
01FF00		LD BC,00FFH	;BC = Byte-Zähler für ;LDIR-Befehl
11017F		LD DE,7F01H	;DE = Hinweisadresse für ;Bestimmungsblock
21007F		LD HL,7F00H	;HL = Hinweisadresse für ;Quellblock
ED B0		LDIR	;A-Registerinhalt in ;Speicherstellen 7F00 – ;7FFF laden

Objektcode		Quellcode	Bemerkungen
010001	CHECK:	LD BC,0100H	;prüfen, ob obige Eingabe ;funktioniert hat, ;BC = Byte-Zählung
21007F		LD HL,7F00H	;HL = Hinweisadresse für ;zu prüfende Speicherstelle
EDA1	NXTLOC:	CPI	;(HL) mit Inhalt von A ;vergleichen
200B		JR NZ, ERROR	;Abweichung zeigt ;Fehler an
E23E01		JP PO,NEXXT	;Paritäts-Merkbit = 0 zeigt ;an: BC = 0000 ;zum nächsten Test-Byte ;übergehen (INC A)
18F7		JR NXTLOC	;Übereinstimmung und BC ;nicht = 0000, zur näch- ;sten Stelle übergehen
FEFF	NEXXT:	CP OFFH	;prüfen ob A = FF
20DE		JR NZ, LOOP4	;falls nicht, nächstes Byte ;prüfen
1820		JR END	;falls ja, Test beendet
08	ERROR	EX AF,AF'	;Fehler-Byte unter Ver- ;wendung von zwei ;Routinen vom Betriebs- ;system des Nanocom- ;puters® anzeigen
3E70		LD A,70H	
08		EX AF, AF'	
3EE0		LD A,0E0H	
32E50F		LD (ADDH),A	;"E" in äußerste linke ;Stelle laden
2B		DEC HL	;HL = Hinweis auf Stelle
7D		LD A,L	
32E20F		LD (DATAL),A	
7C		LD A,H	
32E30F		LD (DATAH),A	
21B90F		LD HL,LEDL	
11E50F		LD DE,ADDH	
CD7CFA		CALL CONVDI	
CD09F9	ERRLP:	CALL DISPL	
18FB		JR ERRLP	
08	END:	EX AF,AF'	;Display Fs, wenn Test ok
3E00		LD A, 00H	
08		EX AF,AF'	
3EFF		LD A,0FFH	
32E50F		LD (ADDH),A	
32E40F		LD (ADDL),A	
32E30F		LD (DATAH),A	
32E20F		LD (DATAL),A	
21890F		LD HL, LEDL	
11E50F		LD DE, ADDH	
CD7CFA		CALL CONVDI	
CD09F9	OK:	CALL DISPL	
18FB		JR OK	

Tabelle 3-11. T54LS04/T74LS04 Eigenschaften.

HEX INVERTER



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS04X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS04X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 µA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			1.2	2.4	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			3.6	6.6	mA	V_{CC} = MAX, Inputs Open

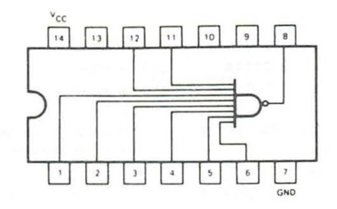
AC CHARACTERISTICS: T_A = 25°C (See Page 4-50 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	5.0	10	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	5.0	10	ns	C_L = 15 pF

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tabelle 3-12. T 54LS30/T74LS30 Eigenschaften.

8-INPUT NAND GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS30X	4 5 V	5 0 V	.5 5 V	-55°C to 125°C
T74LS30X	4 75 V	5 0 V	5 25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2 0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0 7	V	Guaranteed Input LOW Voltage
		74			0 8		
V_{CD}	Input Clamp Diode Voltage			-0 65	-1 5	V	V_{CC} = MIN, I_{IN} = -18 mA
V_{OH}	Output HIGH Voltage	54	2 5	3 4		V	V_{CC} = MIN, I_{OH} = -400 µA, V_{IN} = V_{IL}
		74	2 7	3 4			
V_{OL}	Output LOW Voltage	54, 74		0.25	0 4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0 35	0 5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1 0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0 1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				0.4	mA	V_{CC} = MAX, V_{IN} = 0 4 V
I_{OS}	Output Short Circuit Current (Note 3)		-15		-100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			0 35	0 5	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			0 6	1 1	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C (See Page 4-50 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output		7.0	12	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output		13	20	ns	C_L = 15 pF

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tabelle 3-13. 2101A Eigenschaften. Statisches 256 × 4 Bit-RAM.

2101A-2	250 ns Max.
2101A	350 ns Max.
2101A-4	450 ns Max.

- **256 x 4 Organization to Meet Needs for Small System Memories**
- **Single +5V Supply Voltage**
- **Directly TTL Compatible: All Inputs and Output**
- **Statis MOS: No Clocks or Refreshing Required**
- **Simple Memory Expansion: Chip Enable Input**
- **Inputs Protected: All Inputs Have Protection Against Static Charge**
- **Low Cost Packaging: 22 Pin Plastic Dual In-Line Configuration**
- **Low Power: Typically 150 mW**
- **Three-State Output: OR-Tie Capability**
- **Output Disable Provided for Ease of Use in Common Data Bus Systems**

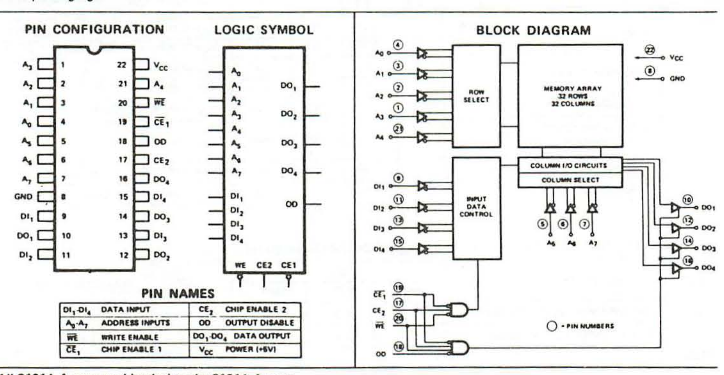
The Intel® 2101A is a 256 word by 4-bit static random access memory element using N-channel MOS devices integrated on a monolithic array. It uses fully DC stable (static) circuitry and therefore requires no clocks or refreshing to operate. The data is read out nondestructively and has the same polarity as the input data.

The 2101A is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives.

It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. Two chip-enables allow easy selection of an individual package when outputs are OR-tied. An output disable is provided so that data inputs and outputs can be tied for common I/O systems. The output disable function eliminates the need for bi-directional logic in a common I/O system.

The Intel® 2101A is fabricated with N-channel silicon gate technology. This technology allows the design and production of high performance, easy-to-use MOS circuits and provides a higher functional density on a monolithic chip than either conventional MOS technology or P-channel silicon gate technology.

Intel's silicon gate technology also provides excellent protection against contamination. This permits the use of low cost plastic packaging.



DI_1-DI_4	DATA INPUT	CE_2	CHIP ENABLE 2
A_0-A_7	ADDRESS INPUTS	OD	OUTPUT DISABLE
$\overline{WE}$	WRITE ENABLE	DO_1-DO_4	DATA OUTPUT
$\overline{CE}_1$	CHIP ENABLE 1	V_{CC}	POWER (+5V)

*All 8101A-4 specs are identical to the 2101A-4 specs.

Tabelle 3-13

2101A FAMILY

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . -10°C to 80°C
Storage Temperature . . . . -65°C to +150°C
Voltage On Any Pin
With Respect to Ground . . . . -0.5V to +7V
Power Dissipation . . . . 1 Watt

**COMMENT:*

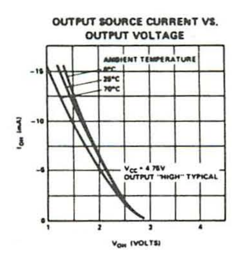
Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

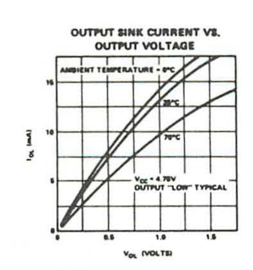
D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified.

Symbol	Parameter		Min.	Typ.[1]	Max.	Unit	Test Conditions
I_{LI}	Input Current			1	10	μA	V_{IN} = 0 to 5.25V
I_{LOH}	Data Output Leakage Current			1	10	μA	Output Disabled, V_{OUT}=4.0V
I_{LOL}	Data Output Leakage Current			-1	-10	μA	Output Disabled, V_{OUT}=0.45V
I_{CC1}	Power Supply Current	2101A, 2101A-4		35	55	mA	V_{IN} = 5.25V, I_O = 0mA
		2101A-2		45	65		T_A = 25°C
I_{CC2}	Power Supply Current	2101A, 2101A-4			60	mA	V_{IN} = 5.25V, I_O = 0mA
		2101A-2			70		T_A = 0°C
V_{IL}	Input "Low" Voltage		-0.5		+0.8	V	
V_{IH}	Input "High" Voltage		2.0		V_{CC}	V	
V_{OL}	Output "Low" Voltage				+0.45	V	I_{OL} = 2.0mA
V_{OH}	Output "High" Voltage	2101A, 2101A-2	2.4			V	I_{OH} = -200μA
		2101A-4	2.4			V	I_{OH} = -150μA

TYPICAL D.C. CHARACTERISTICS





NOTES: 1. Typical values are for T_A = 25°C and nominal supply voltage.

Tabelle 3-13

A.C. CHARACTERISTICS FOR 2101A-2 (250 ns ACCESS TIME)

READ CYCLE T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{RC}	Read Cycle	250			ns	t_r, t_f = 20ns Input Levels = 0.8V or 2.0V Timing Reference = 1.5V Load = 1 TTL Gate and C_L = 100pF.
t_A	Access Time			250	ns	
t_{CO}	Chip Enable To Output			180	ns	
t_{OD}	Output Disable To Output			130	ns	
t_{DF} [3]	Data Output to High Z State	0		180	ns	
t_{OH}	Previous Read Data Valid after change of Address	40			ns	

WRITE CYCLE

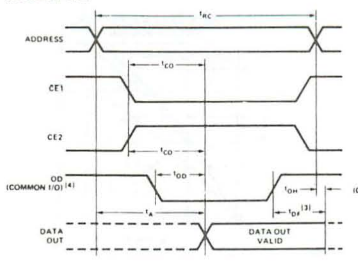
Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{WC}	Write Cycle	170			ns	t_r, t_f = 20ns Input Levels = 0.8V or 2.0V Timing Reference = 1.5V Load = 1 TTL Gate and C_L = 100pF.
t_{AW}	Write Delay	20			ns	
t_{CW}	Chip Enable To Write	150			ns	
t_{DW}	Data Setup	150			ns	
t_{DH}	Data Hold	0			ns	
t_{WP}	Write Pulse	150			ns	
t_{WR}	Write Recovery	0			ns	
t_{DS}	Output Disable Setup	20			ns	

CAPACITANCE [2] T_A = 25°C, f = 1MHz

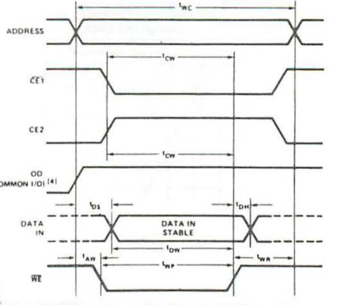
Symbol	Test	Limits (pF)	
		Typ.[1]	Max.
C_{IN}	Input Capacitance (All Input Pins) V_{IN} = 0V	4	8
C_{OUT}	Output Capacitance V_{OUT} = 0V	8	12

WAVEFORMS

READ CYCLE



WRITE CYCLE



NOTES
1. Typical values are for T_A = 25°C and nominal supply voltage.
2. This parameter is periodically sampled and is not 100% tested.
3. t_{DF} is with respect to the trailing edge of $\overline{CE_1}$, CE_2, or OD, whichever occurs first.
4. OD should be tied low for separate I/O operation.

Tabelle 3-13

2101A (350 ns ACCESS TIME)
A.C. CHARACTERISTICS

READ CYCLE T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{RC}	Read Cycle	350			ns	
t_A	Access Time			350	ns	t_r, t_f = 20ns
t_{CO}	Chip Enable To Output			240	ns	Input Levels = 0.8V or 2.0V
t_{OD}	Output Disable To Output			180	ns	Timing Reference = 1.5V
t_{DF} [2]	Data Output to High Z State	0		150	ns	Load = 1 TTL Gate
t_{OH}	Previous Read Data Valid after change of Address	40			ns	and C_L = 100pF.

WRITE CYCLE

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{WC}	Write Cycle	220			ns	
t_{AW}	Write Delay	20			ns	t_r, t_f = 20ns
t_{CW}	Chip Enable To Write	200			ns	Input Levels = 0.8V or 2.0V
t_{DW}	Data Setup	200			ns	Timing Reference = 1.5V
t_{DH}	Data Hold	0			ns	Load = 1 TTL Gate
t_{WP}	Write Pulse	200			ns	and C_L = 100pF.
t_{WR}	Write Recovery	0			ns	
t_{DS}	Output Disable Setup	20			ns	

2101A-4 (450 ns ACCESS TIME)
A.C. CHARACTERISTICS

READ CYCLE T_A = 0°C to 70°C, V_{CC} = 5V ±5%, unless otherwise specified.

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{RC}	Read Cycle	450			ns	
t_A	Access Time			450	ns	t_r, t_f = 20ns
t_{CO}	Chip Enable To Output			310	ns	Input Levels = 0.8V or 2.0V
t_{OD}	Output Disable To Output			250	ns	Timing Reference = 1.5V
t_{DF} [2]	Data Output to High Z State	0		200	ns	Load = 1 TTL Gate
t_{OH}	Previous Read Data Valid after change of Address	40			ns	and C_L = 100pF.

WRITE CYCLE

Symbol	Parameter	Min.	Typ.[1]	Max.	Unit	Test Conditions
t_{WC}	Write Cycle	270			ns	
t_{AW}	Write Delay	20			ns	t_r, t_f = 20ns
t_{CW}	Chip Enable To Write	250			ns	Input Levels = 0.8V or 2.0V
t_{DW}	Data Setup	250			ns	Timing Reference = 1.5V
t_{DH}	Data Hold	0			ns	Load = 1 TTL Gate
t_{WP}	Write Pulse	250			ns	and C_L = 100pF.
t_{WR}	Write Recovery	0			ns	
t_{DS}	Output Disable Setup	20			ns	

NOTES: 1. Typical values are for T_A = 25°C and nominal supply voltage.
2. t_{DF} is with respect to the trailing edge of $\overline{CE_1}$, CE_2, or OD, whichever occurs first.

1. Schritt

In diesem Versuch werden Sie mit der Bedienung eines MOS-Geräts vertraut gemacht, und zwar mit den RAM-ICs 2101A. Um eine Beschädigung der ICs soweit wie möglich zu vermeiden, lesen und befolgen Sie bitte die im Anhang vorgeschlagenen Maßnahmen zur Handhabung von MOS-Geräten.

Mit der Schaltung aus Bild 3-36 erweitern Sie den Schreib/Lesespeicher des Nanocomputers® um 256 Bytes. Lesen Sie die folgenden Eigenschaften des RAM-ICs 2101A sorgfältig durch und beschäftigen Sie sich ausführlich mit der Schaltung aus Bild 3-36.

a) Das IC enthält 256 4-Bit-Speicherworte und hat acht Adressen-Eingänge, die bestimmen, welches der 256 ($=2^8$) 4-Bit-Worte Zugriff hat. Die beiden ICs 2101A arbeiten zusammen, d.h., eine 8-Bit-Adresse hat Zugriff zu einem Byte (das sind insgesamt 8 Bit), je IC 4 Bit.

b) Das IC 2101A hat zwei Sätze zu je vier Anschlußstifte für den Datenaustausch. Ein Satz (vier Leitungen) ist streng für INPUT-Daten bestimmt, während der zweite Satz ausschließlich für OUTPUT-Daten vorgesehen ist. Die INPUT-Daten werden bei einem Speicher-Schreibzyklus von der CPU in den Speicher eingeschrieben. Die OUTPUT-Daten liest die CPU bei einem Speicher-Lesezyklus in den Speicher ein. Maximal einer der zwei Pinsätze kann zu einem gegebenen Zeitpunkt aktiv sein.

c) Die Aktivierung und Reaktivierung der zwei Pinsätze DIn und DOn (n = 1, 2, 3, oder 4) kontrollieren vier Steuerpins: $\overline{CE1}$, $\overline{CE2}$, $\overline{WE}$, und OD. CE ist die Abkürzung für Chip Enable (Chip-Freigabe), WE für Write Enable (Schreib-Freigabe) und OD für Output Enable (Ausgangs-Freigabe). Wie aus dem Blockschaltbild in Tabelle 3-13 hervorgeht, ist folgendes offensichtlich:

   1) Die Eingabe-Datensteuerung aktiviert die Eingabe-Datenpuffer nur dann, wenn $\overline{CE1}$, $\overline{CE2}$ und $\overline{WE}$ zusammen aktiv sind. In dem Schaltbild der Speicherschaltung für diesen Versuch ist $\overline{CE1}$ am Ausgang des NAND-Gatters 74LS30 mit acht Eingängen angeschlossen, CE2 an der Negation des low-aktiven Signals $\overline{BMREQ}$ (Ausgang des Inverters 74LS04) und $\overline{WE}$ am $\overline{DBWR}$-Signal. Alle drei Signale sind nur dann aktiv, wenn ein Speicher-Schreibzyklus ausgeführt wird und auf die obere Hälfte des Adreß-BUS der Hexcode 7F liegt.

   2) Die Daten-Ausgabepuffer werden nur dann aktiviert, wenn $\overline{CE1}$ und $\overline{CE2}$ aktiv und $\overline{WE}$ bzw. OD inaktiv sind. Da OD an $\overline{BRD}$ angeschlossen ist, wird OD nur dann nicht aktiv, wenn $\overline{BRD}$ aktiv ist. Folglich sind die Daten-Ausgabepuffer nur dann frei, wenn ein Speicher-Lesezyklus ausgeführt wird (also $\overline{WE}$=$\overline{DBWR}$ nicht aktiv) und die obere Hälfte des Adreß-BUS den Hexcode 7F führt.

2. Schritt

Bauen Sie jetzt die Schaltung aus Bild 3-36 auf. Wegen der Komplexität nachstehend ein paar Hinweise, um den Drahtwirrwarr etwas zu ordnen:

a) Lage der ICs auf der SK-10 Lochrasterplatte: Alle Adreß- und Daten-BUS-Leitungen müssen zwischen der RAM-Schaltung auf dem SK-10-Platte und den Anschlüssen B und C miteinander verbunden sein. Es ist am besten, die 2101A-ICs und das 74LS30-IC möglichst nahe an

diese Anschlüsse heranzubringen. Die beiden 2101A-ICs sollten nebeneinander liegen und das 74LS30 links oder rechts daneben. Die Anordnung des 74LS04 ist unproblematisch.

b) Zunächst die ICs mit der Versorgungsspannung und Masse verbinden. Dabei muß die Stromversorgung vom Netz getrennt sein!
Mehrere Signale werden jedem 2101A-IC zugeführt. Es sind dies die Signale auf der unteren Adreß-BUS-Hälfte BA0 . . . BA7, $\overline{BMREQ}$, $\overline{BRD}$ und $\overline{DBWR}$. Verbinden Sie die entsprechenden Pinpaare mit möglichst kurzen Drähten.

c) Die Daten-In-Pins (DI1-DI4) an den beiden 2101A-ICs sind mit ihren entsprechenden Daten-Out-Stiften (DO1 . . . DO4) verbunden! Verdrahten Sie diese Paare als nächstes, ebenfalls mit kurzen Drähten.

d) Den Adreß-, Daten-BUS und die Steuersignale der Anschlüsse B und C mit den entsprechenden Anschlußstiften der ICs 2101A, 74LS30 und 74LS04 verbinden. Es ist vorteilhaft, für den Adreß- und den Daten-BUS Drähte mit anderen Farben zu benutzen.

3. Schritt

Beim nächsten Schritt soll versucht werden, Daten, die Sie Ihrem Nanocomputer® hinzugefügt haben, in den Speicher einzulesen bzw. aus dem Speicher auszuschreiben. Bevor Sie dies jedoch können, sind einige grundsätzliche Punkte zu beachten.

a) *Für welchen Adressenbereich ist der Speicher ansprechbar?* Diese Frage können Sie durch einen Blick auf das Schaltbild 3-36 beantworten. Die untere Hälfte der Adreß-BUS-Leitungen (BA0 . . . BA7) werden den Speicher-ICs direkt zugeführt. Diese Leitungen tragen die niedrigwertigsten acht Bits aller Adressenstellen zu denen die Z-80-CPU Zugriff hat. Der Bereich der möglichen Werte für diese acht Bits reicht von 00 bis FF. Das sind genau 256 Möglichkeiten, von denen keine dem Zufahll überlassen ist. Wie steht es mit den oberen 8 Adressenleitungen BA8 . . . BA15? Sie gelangen auf die 8 Eingänge des NAND-Gatters 74LS30, wobei das Signal BA15 zuerst über einen Inverter geführt wird. Der Ausgang des NAND-Gatters ist also nur dann logisch 0, wenn auf den Adreßleitungen BA8 . . . BA15 folgende Situation eintritt

BA–15	14	13	12	11	10	9	8
0	1	1	1	1	1	1	1

Mit anderen Worten, der Ausgang des 74LS30, der zusammen mit dem Signal $\overline{BMREQ}$ als Chip-Freigabe dient, ist nur dann aktiv, wenn eine Adresse der Form 7FXX auf dem Adreß-BUS ist (XX = nicht beachten). Die Antwort auf die Eingangsfrage lautet demnach: Mit allen Adressen im Bereich 7F00 bis 7FFF wird der Speicher 2101A angewählt.

b) *Wie kann man mit dem Speicher 2101A korrespondieren?* Das ist relativ leicht. Benutzen Sie dazu das Betriebssystem des Nanocomputers®. Bringen Sie die Auswahl-LED in Stellung MEM, laden die Adresse 7F00 und betätigen die INC-Taste.

Versuchen Sie die Speicherstellen 7F00 bis 7F0E zu lesen. Sie beobachten

dabei eine Vielzahl verschiedener Bytes. Wenn alle Stellen FF lauten, ist dies ein schlechtes Zeichen für Ihre Schaltung, die nicht richtig funktioniert. Versuchen Sie zunächst verschiedene Stellen mit 00 zu laden. Wie machen Sie das?
Benutzen Sie wiederum das Betriebssystem. Geben Sie lediglich die zu ändernde Adresse ein, drücken die Taste LA, geben die neuen Daten ein und drücken die Taste ST. Prüfen Sie, ob die neuen Daten richtig eingespeichert sind. Wenn die Stelle immer noch FF enthält, haben Sie zweifellos eine fehlerhafte Speicherschaltung. Nachstehend einige Vorschläge zur Fehlerbeseitigung in einer Problemschaltung:

a) Trennen Sie den Dekoderteil der Schaltung, nämlich den 74LS30, von der übrigen Schaltung ab, und prüfen Sie die Funktion.
   Das können Sie mit einem Impuls-Auffänger machen, eine Schaltung ähnlich dem Bit-Auffänger im vorigen Versuch. Um festzustellen, ob der Ausgang des 74LS30 nur aktiv ist, wenn 7F auf der oberen Hälfte des Adreß-BUS erscheint, fertigen Sie sich eine Impuls-Auffang-Schaltung an (Bild 3-37).

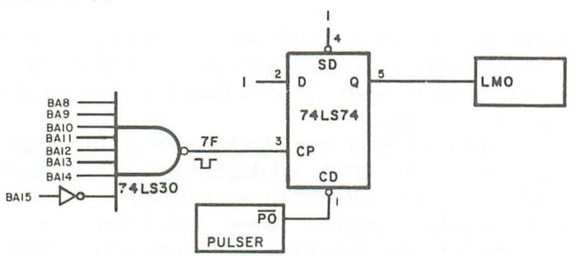


Bild 3-37. Impuls-Auffang-Schaltung. Die Aufgabe des Impulsgebers PULSER können Sie selbst übernehmen.

   Falls die LED LMO leuchtet, verbinden Sie den CP-Eingang des 73LS74 mehrmals hintereinander mit Masse, bis sie erlischt. Betätigen Sie beim Nanocomputer® die RESET-Taste und geben Sie anschließend einige Speicherstellen ein. Ausgenommen ist der Bereich 7F00 bis 7FFF. Dabei darf LMO *nicht* aufleuchten, weil der Ausgang des 74LS30 seinen logischen Zustand "1" *nicht* wechselt. Anschließend testen Sie einige Speicherstellen im Bereich 7F00 bis 7FFF. Die LED LMO muß bei jeder Adresse innerhalb des angegebenen Bereiches aufleuchten. Das ist deswegen der Fall, weil das Betriebssystem des Nanocomputers® in diesem Bereich ein 7F auf die obere Hälfte des Adreß-BUS legt. Wenn der Dekoderteil der Schaltung so funktioniert, ist er in Ordnung.
b) Etwaige Fehler an der übrigen Schaltung lassen sich mit ähnlichen Techniken beseitigen.

4. Schritt

Laden Sie das Programm MEM1 in den Speicher des Nanocomputers® (Startadresse z.B. 0100). Dieses Programm ist ein sehr einfacher Speichertest. Sollte Ihr neu eingebauter Speicher diesen Test nicht bestehen, liegt ein Fehler vor, evtl. defekte Speicherzellen. Besteht der Speicher aber diesen Test, dann wissen Sie immer noch *nicht,* ob er in allen Bereichen

fehlerfrei arbeitet. Die Wissenschaft der Speicherprüfung übersteigt den Rahmen dieses Buches. Die entwickelten Funktionsprüfungen für Speicher sind teilweise sehr umfangreich und würden Monate, wenn nicht Jahre dauern.
Ein Flußdiagramm für das Speicher-Testprogramm, beginnend bei der Speicherstelle MEM1, ist in Bild 3-38 dargestellt. Im wesentlichen lädt das Programm alle 256 Bytes von 7F00 bis 7FFF mit allen 8-Bit-Mustern, die möglich sind und prüft, ob der Ladevorgang erfolgreich war. Die

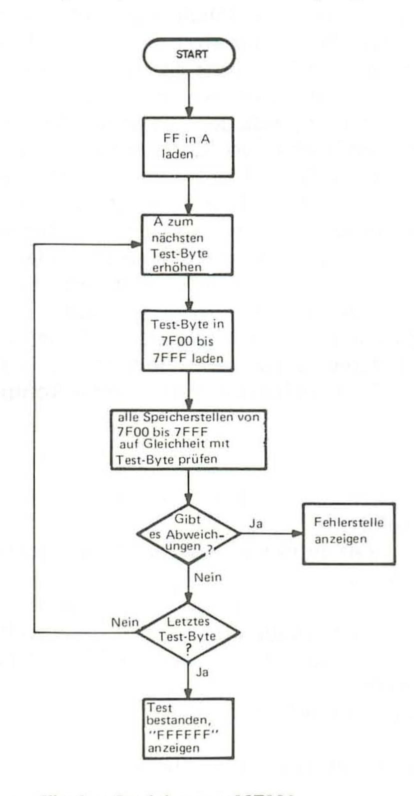


Bild 3-38. Flußdiagramm für den Speichertest MEM1.

äußere Schleife ändert das Bit-Muster, während die innere Schleife die Befehle LDIR und CPI benutzt, um den Ladevorgang auszuführen bzw. zu prüfen. Ein umfangreicherer Speichertest ändert nur jeweils *ein* Byte und prüft, ob sich auch eine andere Speicherstelle verändert hat. Auch Änderungen in der Reihenfolge, in der Bytes eingelesen oder ausgeschrieben werden, wären Teil eines umfangreicheren Speichertests. Der MEM1-Test schreibt bzw. liest die Speicherstellen immer in aufsteigender Reihenfolge.
Beim Speichertest MEM1 gibt es zwei mögliche Ergebnisse: bestanden oder nicht bestanden. Besteht der Speicher den Test, lautet die Anzeige

FFFFFF. Arbeitet der Speicher fehlerhaft, lautet die Anzeige E XXXX (XXXX = Adresse der fehlerhaften Speicherstelle).

Führen Sie das Speicher-Testprogramm MEM1 durch. Was fällt Ihnen auf? Zunächst ist die Anzeige sekundenlang dunkel, um dann aufzuleuchten und FFFFFF anzuzeigen. Das bedeutet, der Speicher hat den Test bestanden.
Hat der Speicher den Test nicht bestanden gibt es keine automatische Anzeige. Drücken Sie die Break-Taste, damit die Steuerung zum Betriebssystem zurückkehrt, ohne den Inhalt der Register zu verändern. Das Display zeigt den Inhalt der fehlerhaften Speicherstelle an.
Die Fehlernachricht des Speichertests entsteht, weil der Inhalt der fehlerhaften Speicherstelle hätte anders sein müssen. Um festzustellen, wie der Inhalt hätte lauten müssen, schauen Sie entweder nach dem Inhalt der Speicherstelle, die unmittelbar vor der Fehlerstelle liegt oder nach dem Inhalt des Akkumulators. Schreiben Sie mit Hilfe des Betriebssystems des Nanocomputers® verschiedene Test-Bytes in die defekte Speicherstelle. Unter Umständen finden Sie zwischen dem, was Sie lesen und schreiben Diskrepanzen, weil die Störung durch eine besondere Kombination von Ereignissen verursacht sein kann. Vielleicht liegt auch ein sogenannter *intermittierender Fehler* vor, der unvorhersehbar ist. So kann z.B. bei wiederholter Ausführung des Speichertests das Ergebnis mehrmals positiv sein als auch Zufallsfehler aufweisen. Wie bereits erwähnt, können Suche und Behebung von Speicherfehlern eine äußerst komplizierte Angelegenheit sein.

5. Schritt

Bauen Sie nun absichtlich einen Fehler an einer bestimmten Speicherstelle ein, und zwar wie folgt:

a) Legen Sie einen Haltepunkt direkt hinter dem LDIR-Befehl am Speicherplatz CHECK an.
b) Programm ausführen. Ist der Haltepukt gefunden, laden Sie ein Byte – außer 00 – in die Speicherstelle 7F13. Auf diese Weise manipulieren Sie den Speicher zwischen dem Laden und der Check-Operation.
c) Haltepukt entfernen.
d) Setzen Sie das Programm fort.

Auf der Anzeige erscheint folgende Fehlernachricht:

E 7F13

Weitere Fehlernachrichten erhält man, indem die obigen Schritte an einer anderen Speicherstelle wiederholt werden.

6. Schritt

Angenommen, Sie möchten die Speicherstellen der 256 RAM-Bytes in den beiden ICs 2101A in einen anderen Speicherbereich wie z.B. BF00 . . . BFFF verändern. Wie läßt sich dies verwirklichen? Falls Sie eine Idee haben, probieren Sie!
Der einfachste Weg, den Speicher RAM-2101A Bereich BF00 . . .BFFF zu verschieben, ist der Austausch der Anschlüsse BA14 und BA15 am NAND-

Gatter 74LS30 (Bild 3-39).

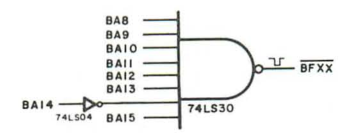


Bild 3-39. Durch den Anschluß des Inverters an BA14 ändert sich der Speicherbereich in BF00 . . . BFFF.

Die Freigabe durch das NAND-Gatter 74LS30 ist dann nur bei folgendem Bitmuster möglich:

BA–15	14	13	12	11	10	9	8
1	0	1	1	1	1	1	1

7. Schritt

Benutzen Sie die Invertierer im 74LS04 und verschieben den RAM-Speicher 2101A in den Bereich

3F00 . . . 3FFF
und
C000 . . .C0FF

Zeichnen Sie das Schaltbild zum Verdrahten des NAND-Gatters 74LS30 und der Invertierer 74LS04 für jeden der vorigen Bereiche. Die Richtigkeit der Schaltbilder können Sie durch Ausprobieren kontrollieren.
NICHT DIE SPEICHERSCHALTUNG ABBAUEN, SIE WIRD FÜR DEN NÄCHSTEN VERSUCH BENÖTIGT!

VERSUCH NR. 5

Dieser Versuch verdeutlicht den Unterschied zwischen einem statischen und einem dynamischen RAM und seine Bedeutung im Hinblick auf die Verwendung von Wartezuständen.
Dieser Versuch benutzt die Software und Hardware aus Versuch Nr. 4 (siehe Bild 3-36).

Programm XFER

Objekt-Code		**Quell-Code**	**Bemerkungen**
		NAME XFER	
016600	XFER:	LD BC,0K+5H-MEM1	;Zahl der Bytes im MEM1- ;Programm bestimmt den ;Aufbau des LDIR 0K+5H- ;MEM1
11007F		LD DE,7F00H	;statisches RAM ist ;Bestimmungsort
211E01		LD HL,MEM1	;Quellblock ist MEM1- ;Programm
EDB0		LDIR	;Do it (mach es)
FF		RST 38H	;Steuerung zum Betriebs- ;system des Nanocom- ;puters® zurückgeben

1. Schritt

Das 256-Byte-2101A-R/W-RAM ist ein STATISCHER Lese/Schreib-Speicher mit wahlfreiem Zugriff. Das R/W-RAM im Bereich 0000 . . . 0FFF des Nanocomputers® ist ein *dynamischer* Speicher. Es gibt mehrere wichtige Unterschiede zwischen einem *statischen* und einem *dynamischen* R/W-RAM. Die Speichereinheiten in statischen R/W-RAMs sind Flipflops. Wenn Flipflops einmal im Zustand logisch 0 oder 1 sind, behalten sie ihren Wert, solange die Versorgungsspannung anliegt und kein Änderungsimpuls zugeführt wird.

Dagegen benötigen dynamische R/W-RAMs Kondensatoren als Grund-Speichereinheit. Die Ladung des Kondensators bestimmt, ob sich ein Speicher-Bit im Zustand logisch 0 oder 1 befindet. Leider kann ein Kondensator in wenigen Millisekunden soviel Ladung verlieren, daß er von einem logischen Zustand in den anderen wechselt. Darum müssen dynamische Speicher im Abstand von wenigen Millisekunden eine Refresh-Operation ausführen, damit ihre Kondensatoren aufgeladen werden. Obwohl es deshalb etwas schwieriger ist mit dynamischen R/W-RAMs zu arbeiten als mit statischen, weisen Sie doch einige Vorteile auf. Sie verbrauchen weniger Strom, haben eine höhere Packungsdichte und kosten

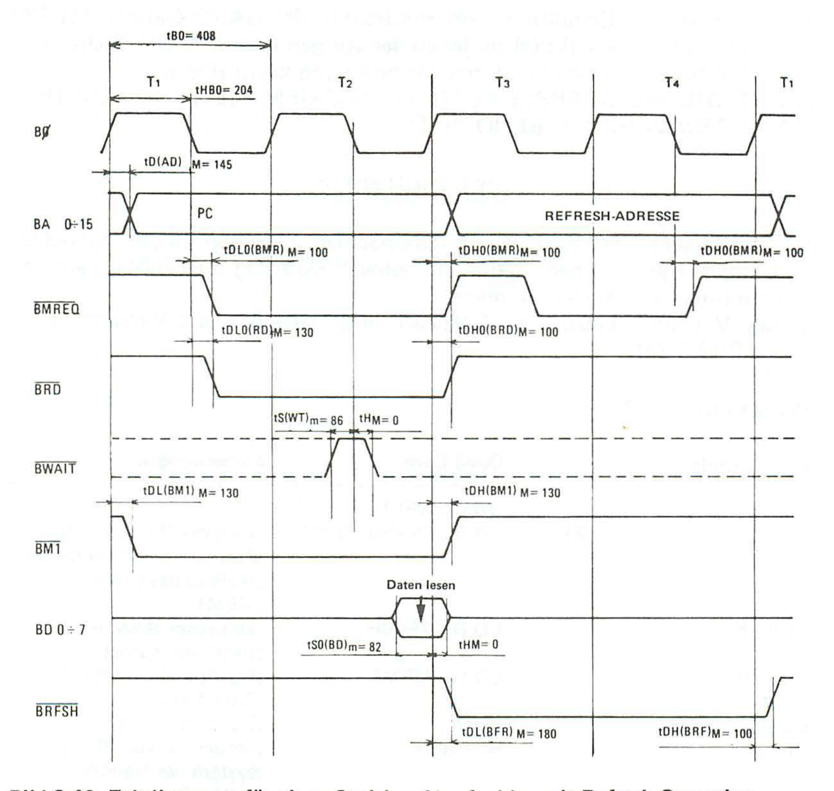


Bild 3-40. Zeitdiagramm für einen Speicher-Abrufzyklus mit Refresh-Operation.

weniger. So haben also beide Arten ihre Vorteile und werden bei den verschiedensten Anwendungen häufig eingesetzt. Da sowohl der statische als auch der dynamische Speicher seine Daten verliert, sobald die Stromzufuhr ausfällt, nennt man sie *flüchtige* Speicher. Der EPROM-Speicher, der das Betriebssystem des Nanocomputers® enthält, ist *nicht* flüchtig.
Wie in Kapitel 1 bereits erwähnt, erzeugt die Z-80-CPU die erforderlichen Signale, um die Refresh-Aufgaben bei den meisten dynamischen RAMs auszuführen. Refresh-Signale stellen bei Abruf-(M1)-Zyklen einen Ausgang dar und benutzen den Adreß-BUS sowie die Signale $\overline{\text{BMREQ}}$ und $\overline{\text{BRFSH}}$. In Bild 3-40 ist ein Speicher-Abrufzyklus dargestellt. Da dynamische R/W-RAMs im Abstand von wenigen Millisekunden Refresh-Operationen benötigen, muß die Z-80-CPU mindestens ebenso oft Speicher-Abrufzyklen ausführen, damit sie den Inhalt nicht "verlieren". Die meisten Z-80-Befehle benötigen weit weniger als eine Millisekunde zur Ausführung, daher stehen beim normalen Programmablauf genügend Speicher-Refreshzyklen zur Verfügung, um den Speicherinhalt zu regenerieren. Befehle, die möglicherweise Schwierigkeiten bereiten können, sind Blocklade-, Vergleichs- und I/O-Befehle wie z.B. LDDR, CPIR usw. Diese Befehle sind konzipiert, um während ihrer (möglicherweise langen) Ausführung Speicher-Refresh-Operationen nach jeder Datenübertragung durchzuführen. Deshalb verursachen auch Blockverschiebungen von einigen tausend Bytes bei dynamischen Speichern keine Probleme. Das können Sie selbst ausprobieren. Seien Sie aber vorsichtig, da es mehrere "Fallgruben" gibt. Überschreiben Sie nicht ihr eigenes Programm, und dezimieren Sie nicht den Stapel- und Datenraum des Operationssystems, Speicherstellen 0EF0 . . . 0FFF.

2. Schritt

Es gibt noch einen Befehl, der Schwierigkeiten bereiten kann: HALT. Die Entwickler der Z-80-CPU haben vorgesorgt, indem die Ausführung eines HALT-Befehls als die Ausführung eines endlosen Stomes von NOP-Befehlen definiert ist. Da ein NOP-Befehl zu seiner Ausführung vier T-Zyklen benötigt, folgen alle vier T-Zyklen Refresh-Operationen, wenn sich die Z-80 in einem HALT-Zustand befindet. Ein T-Zyklus dauert ca. 400 ns. 4 X 400 ns = 1,6 Mikrosekunden – weniger als eine Millisekunde.

Um experimentell zu kontrolliern, ob der HALT-Befehl einen dynamischen Speicher nicht beeinflußt, laden Sie normale Byte-Muster in mehrere Speicherstellen, beginnend bei 0300. Laden Sie in die Speicherstelle 0220 00 (Maschinencode für den NOP-Befehl) und in die Speicherstelle 0221 76 (HALT). Führen Sie das Programm, beginnend bei der Speicherstelle 0220, durch Drücken der Taste GO aus. Lassen Sie die CPU einige Sekunden lang im HALT-Zustand, und erzeugen Sie durch Drücken der BREAK-Taste ein NMI, um den Haltezustand zu verlassen. Dann prüfen Sie, ob sich die ab 0300 geladenen Bytes verändert haben. Dies nicht der Fall sein.

3. Schritt

Bauen Sie die Speicherschaltung aus Versuch 4 aus (Bild 3-36): Der statische 2101A-RAM muß wieder bei 7F00-7FFF sein. Geben Sie das Speicher-Testprogramm ein, beginnend bei der Speicherstelle MEM1.

Führen Sie das Programm XFER aus, indem Sie bei der Speicherstelle XFER beginnen, um die Speicher-Testroutine MEM1 unverändert in das statische RAM zu übertragen. Prüfen Sie, ob die Übertragung erfolgreich war.

4. Schritt

Wenn die MEM1-Routine sicher im statischen RAM gespeichert ist, verbinden Sie die $\overline{\text{BWAIT}}$-Leitung von der Z-80-CPU kurzzeitig mit Masse. Benutzen Sie dazu einen Draht, *keinen* Schalter oder Impulsgeber, weil die $\overline{\text{BWAIT}}$-Leitung ein offenes Kollektorgatter benötigt, wenn ein IC benutzt wird.
Prüfen Sie mehrere dynamische Speicherstellen, beginnen bei MEM1. Was beobachten Sie?
Das Testprogramm ist vollkommen zerstört! Prüfen Sie jetzt mehrere Speicherstellen im statischen RAM, beginnend bei 7F00. Sie finden eine perfekte Kopie der MEM1-Routine vor.
Der Schritt 4 verdeutlicht einen wichtigen Vorteil eines statischen gegenüber einem dynamischen R/W-RAM. Während die Aktivierung der $\overline{\text{BWAIT}}$-Leitung für mehrere Wartezustände keinen Einfluß auf ein statisches RAM haben, dezimiert sie das dynamische R/W-RAM vollkommen. Die Wartezustände hindern die CPU an der Ausführung eines Speicher-Abrufzyklus. Werden zu viele Speicher-Abrufzyklen ausgelassen, fehlt beim dynamischen R/W-RAM die Auffrischung und somit ist der Inhalt verloren. Solange die $\overline{\text{BWAIT}}$-Leitung mit Masse verbunden ist, bleibt die CPU im Wartezustand und unterläßt die notwendigen Refresh-Operationen. Ein Wartezustand hat einen T-Zyklus von 400 ns. Bei Zehntausend Wartezuständen entsteht zwischen zwei Speicher-Abrufzyklen eine Lücke von 4 ms. Das ist bereits genug, um den Inhalt eines dynamischen Speichers zu zerstören. Bereits bei einer Lücke von 2 oder 3 ms Sekunden ist der Inhalt gefährdet.
Die Aktivierung des $\overline{\text{BBUSRQ}}$-Signals verhindert die Ausführung von Speicher-Abrufzyklen ebenfalls. Das $\overline{\text{BBUSRQ}}$-Signal veranlaßt die Z-80-CPU, den Daten-, Adreß- und Steuer-BUS an eine andere CPU außerhalb des Nanocomputers® abzugeben. Während die externe CPU die BUS-Leitungen des Nanocomputers® benutzt, kann die CPU des Nanocomputers® keine Speicher-Abrufzykeln ausführen, weil dazu der Adreß- und Steuer-BUS erforderlich sind. Das $\overline{\text{BBUSRQ}}$-Signal wird für Vielfach-CPU-Konzepte und direkte Speicherzugriffoperationen (DMA) benutzt. Beide Themen sind in diesem Buch nur kurz erwähnt.

5. Schritt

Da sich der Speichertest jetzt im statischen R/W-RAM befindet, können Sie ihn (mit geringfügigen Änderungen) zur Prüfung des R/W-RAM ihres Nanocomputers® benutzen. Zur Prüfung der Speicherstellen 0100 . . . 01FF nehmen Sie folgende Änderungen vor:

Speicherort	Alter Inhalt	Neuer Inhalt
7E05	7F	01
70FB	7F	01
7F0E	7F	01
7F16	7F	01
7F1D	01	7F

Führen Sie den neuen Speichertest aus, beginnend bei 7F00.

Kapitel 4

BUS-Leitungen, Three-State-Puffer und Z-80-I/O

Es ist aus den vorigen Kapiteln bereits bekannt, daß sich die Z-80-CPU mit der Umwelt über drei BUS-Leitungen in Verbindung setzt, nämlich dem DATEN-BUS, dem ADRESS-BUS und dem STEUER-BUS. Während bisher der Begriff BUS nur kurz erläutert war, geht dieses Kapitel näher auf die BUS-Leitungen ein. Sie werden die Innenstruktur der Z-80-CPU näher kennenlernen und BUS-Techniken untersuchen, die z.Zt. im Mikrocomputerbau und vielen Mikrocomputersystemen aktuell sind.

Nach dem Studium dieses Kapitels werden Sie in der Lage sein:
- den Begriff und das Verb BUS (Sammelschiene – an Sammelschiene anschließen) zu definieren;
- die Eigenschaften eines TRI-STATE® oder Three-State-(Drei-Zustands-)-Puffers zu erklären einschließlich der Daten-, Freigabe/Blockier-Eingänge und des Three-State-Ausgangs;
- eine Wahrheitstabelle für ein Three-State-Gerät zu schreiben;
- den Begriff *offener Kollektorausgang* zu definieren und zu verstehen, wie offene Kollektorausgänge beim Anschluß von Mikrocomputern an Sammelschienen benutzt werden;
- Beispiele für ein einfaches BUS-System zu geben;
- die allgemeinen Eigenschaften von Three-State-ICs wie die Puffer-Treiber 74LS125, 74LS126 und 74LS365 zu beschreiben;
- mehrere auf dem Markt befindliche Three-State-ICs zu nennen;
- *Einggangs-Lastfaktor* und *Ausgangs-Lastfaktor* zu definieren und ihre Bedeutung in I/O-Schaltungen von Mikroprozessoren zu erklären;
- eine einfache Z-80-Ausgangsschaltung zu zeichnen und zu verdrahten;
- eine einfache Z-80-Eingangsschaltung zu zeichnen und zu verdrahten;
- zwischen Gatter-I/O und Speicherplan-I/O bei einem Mikrocomputer auf Z-80-Basis zu unterscheiden.

WAS IST EIN BUS?

Ein BUS ist ein Satz Leitungen, über die digitale Informationen von verschiedenen Quellen zu verschiedenen Bestimmungsorten übertragen werden. Der Hauptzweck von einem BUS ist es, die Zahl der erforderlichen Verbindungswege auf ein Mindestmaß zu beschränken. Daher können zur gleichen Zeit nie mehrere Übertragungen stattfinden. Während einer Übertragung müssen alle anderen mit dem BUS verbundenen Quellen blockiert bleiben. Das Verb BUS bedeutet, daß mehrere Digital-Geräte miteinander

verbunden sind, die entweder digitale Information empfangen oder senden. Dies geschieht über einen Satz von Leitungen, die BUS genannt werden und über den alle Informationen zwischen solchen Geräten fließen.

BUS-Leitungen befinden sich:
- in integrierten Schaltungsgeräten, z.B. der innere Daten-BUS in einem Z-80-Mikroprozessor;
- zwischen integrierten Schaltungen, z.B. der Adreß-Steuer- und bidirektionale Daten-BUS in einem Z-80-Mikrocomputer;
- zwischen Digitalsystemen und Instrumenten, z.B. der Interface-BUS IEEE-488, der heute ein Standard-Interface zwischen Digital-Instrumenten ist.

In der Digital-Elektronik ist das BUS-Konzept wahrscheinlich die wichtigste Idee überhaupt. Ohne die Möglichkeit, Informationswege gemeinsam zu benutzen, stieg bei Digitalgeräten die Zahl der Drahtanschlüsse auf ein Vielfaches an. Gedruckte Schaltungen (Platinen) für Mikro- und Minicomputer wären weitaus komplexer, daher teurer und weniger zuverlässig.

THREE-STATE BUS-VERBINDUNGEN

Wie bereits erwähnt, hat ein BUS die beiden folgenden, sich scheinbar widersprechenden Eigenschaften:
- mehrere Quellen und mehrere Bestimmungsorte über gewöhnliche Leitpfade miteinander zu verbinden;
- zur gleichen Zeit kann immer nur EINE Information übertragen werden. Wenn auf einem BUS zwischen zwei Geräten eine Datenübertragung stattfindet, müssen daher alle unbeteiligten Geräte vom BUS abgetrennt sein.

Ein Beispiel für das BUS-System zeigt Bild 4-1: Die Geräte TX1 und TX2 senden und RX1 sowie RX2 empfangen Information über den BUS. Wenn TX1 Daten an RX1 sendet und TX2 gleichzeitig ein Binärmuster von Nullen und Einsen auf den BUS legt, wird die Nachrichtenübermittlung zwischen TX1 und RX1 verstümmelt. RX1 empfängt wahrscheinlich eine Mischung von TX1- und TX2-Daten. Das Gerät TX2 muß also irgendwie blockiert werden. Eine Alternative wäre, den Drahtanschluß zwischen TX2 und dem BUS physikalisch zu unterbrechen . . . aber das ist keine gangbare Lösung.
Die optimale Lösung für TX2 ist, das Bitmuster am Ausgang zum BUS hin zu blockieren. Das heißt, zwischen Gerät und BUS ein Digital-Gatter zu schalten, daß über einen Enable/Disenable Eingang (Freigabe/Sperren) entweder die Information von TX2 durchläßt oder sperrt. Die Wahrheitstabelle für dieses ideale Gatter zeigt die Tabelle 4-1.
Bisher sind Digital-Geräte als Two-State-Geräte bekannt. Sie sind in der Lage, eine logische 0 zu registrieren, was dem Massepotential entspricht oder eine logische 1, entsprechend dem +5V-Potential. Für BUS-Systeme ist ein dritter, ''getrennter'' Zustand unentbehrlich. Die heute auf dem Markt befindlichen Three-State-Gatter haben folgende Eigenschaften:

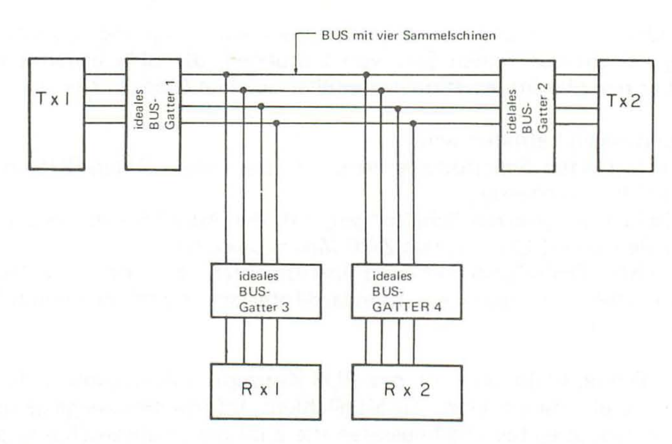


Bild 4-1. Einfaches BUS-System mit vier Geräten und vier Sammelschienen.

Tabelle 4-1. Wahrheitstabelle für ein optimales BUS-Gatter

Eingangsdaten (Ausgangsdaten TX2)	Gattersignal	Ausgangsdaten für BUS
0000	enable (freig.)	0000 (übertragt Daten)
0001	enable (freig.)	0001 (übertragt Daten)
0010	disen. (sperren)	trennt Gerät vom BUS
0011	disen. (sperren)	trennt Gerät vom BUS

- Serie 54/74 TTL-kompatibel;
- bis zu 128 Puffer können an einer gewöhlichen BUS-Leitung angeschlossen werden;
- 12 ns Verzögerung;
- hohe kapazitive Treiberleistung;
- separate Puffersteuerung;
- ermöglichen ein Zusammenschließen von Ausgängen mit Anschluß an eine gewöhnliche BUS-Leitung;
- verbesserte Rechteck-Integrität und Geschwindigkeit über offenen Kollektorausgang;
- verbesserte Übertragung in zwei Richtungen über eine gewöhnliche Leitung, weil die Eingänge des Three-State-Gatter im hochohmigen Zustand nicht die normale Ladung zum Treibergerät darstellen.

Zusammengefaßt hat ein *Three-State-Gatter* drei mögliche Ausgangs-Zustände:

logisch 0
logisch 1
hochohmig (getrennt)

Wenn es sich nicht in hochohmigen Zustand befindet, verhält sich ein Three-State-Gatter wie ein normales TTL-Gatter. Im blockierten Zustand verhält es sich so, als wäre es von der Schaltung getrennt. Alle Three-State-Gatter haben einen Eingangs-Pin, den man Freigabe/Blockier-Eingang

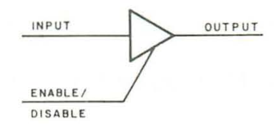


Bild 4-2. Gatter symbol für einen Three-State-Puffer.

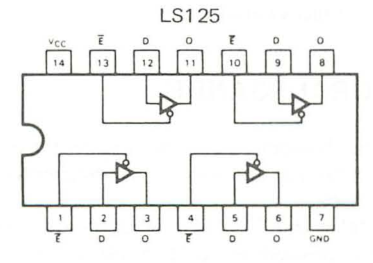
nennt und der bestimmt, ob sich das Gatter im normalen TTL- oder im hochohmigen Zustand befindet. Bild 4-2 zeigt ein Gatter-Symbol sowie die Wahrheitstabelle für einen Three-State-Puffer, der durch eine logische 1 an seinem Enable/Disenable-Eingang (Freigabe/Blockier-Anschluß) den Ausgang freigibt.

THREE-STATE-PUFFER 74LS125 UND 74LS126 MIT VIER PUFFERSTUFEN

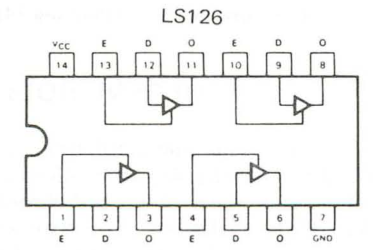
Die Three-State-Puffer 74LS125 und 74LS126 haben jeweils vier Puffer mit einem separaten Freigabe/Blockier-Pin für jeden Puffer. Der einzige Unterschied zwischen den beiden ICs, die Puffer des 74LS125 werden mit logisch 0 und die des 74LS126 mit logisch 1 freigegeben. Die beiden ICs sind in Bild 4-3 dargestellt.
Bei beiden ICs wird die Versorgungsspannung mit Pin 14 und die Masse mit Pin 7 verbunden. Die Pinbelegung der einzelnen Puffer ist wie folgt:

Puffer 1: Eingang bei Pin 2, Ausgang bei Pin 3, Freigabe/Blockierung bei Pin 1
Puffer 2: Eingang bei Pin 5, Ausgang bei Pin 6, Freigabe/Blockierung bei Pin 4

LS125



LS126



TRUTH TABLES

LS125

INPUTS		OUTPUT
Ē	D	
L	L	L
L	H	H
H	X	(Z)

LS126

INPUTS		OUTPUT
E	D	
H	L	L
H	H	H
L	X	(Z)

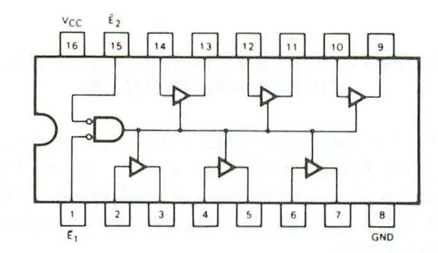
L = LOW-Spannungspegel
H = HIGH-Spannungspegel
X = ohne Bedeutung
(Z) = hochohmig

Bild 4-3. Three-State-Puffer-ICs 74LS125 und 74LS126 mit vier Pufferstufen.

Puffer 3: Eingang bei Pin 9, Ausgang bei Pin 8, Freigabe/Blockierung bei Pin 10
Puffer 4: Eingang bei Pin 12, Ausgang bei Pin 11, Freigabe/Blockierung bei Pin 13

THREE-STATE-PUFFER 74LS365 MIT SECHS PUFFERSTUFEN UND NOR-FREIGABE-GATTER MIT ZWEI EINGÄNGEN

Dieses IC ist zusammen mit seiner Wahrheitstabelle in Bild 4-4 dargestellt. Es enthält sechs Puffer, die gleichzeitig durch den Ausgang eines internen NOR-Gatters mit zwei Eingängen freigegeben werden. Bei Anschlußgeräten zu den externen Z-80-BUS-Leitungen ist es oft nicht erforderlich, jede Leitung getrennt zu steuern, besonders nicht bei der Ein- und Ausgabe von 8-Bit-Bytes. Aus diesem Grunde ist das IC 74LS365 sehr nützlich.



WAHRHEITSTABELLE

INPUTS			OUTPUT
$\bar{E}_1$	$\bar{E}_2$	D	
L	L	L	L
L	L	H	H
H	X	X	(Z)
X	H	X	(Z)

Bild 4-4. Integrierte Pufferschaltung 74LS365 mit sechs Puffern.

OFFENE KOLLEKTORAUSGÄNGE

Eine immer seltener werdende, langsamere, billigere und "geräuschvollere" Möglichkeit, die Ausgänge eines logischen Geräts an einen BUS anzuschließen, stellen die *offenen Kollektorausgänge* dar.
Bei ICs mit offenem Kollektorausgang fehlt der "Pull-Up-Widerstand" in der Ausgangs-Transistorstufe. Dieser Ausgangswiderstand muß der Entwickler beim Schaltungsentwurf einsetzen. Das Fehlen des integrierten Widerstandes macht die Gatter mit offenem Kollektorwiderstand universell verwendbar. Man kann die Gatterausgänge miteinander verbinden und über einen einzigen Widerstand abschließen. So lassen sich z.B. die sechs Inverter des ICs 74LS05 mit offenem Kollektorausgang so zusammenschalten, wie es das Bild 4-5 zeigt.
Die sechs Inverter-Ausgänge des ICs 74LS05 sind unter Verwendung eines 1000 Ohm-Widerstands an +5V zusammengeschlossen. Der Ausgang Q

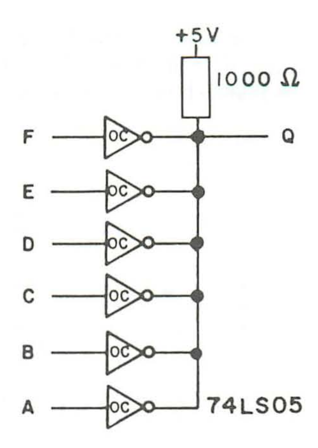


Bild 4-5. Inverter-IC 74LS05 mit sechs offenen Kollektorausgängen, die über einen 1000 Ohm-Widerstand an +5 V zusammengeschaltet sind.

befindet sich nur dann im Zustand logisch 1, wenn ABCDEF = 000000. Befindet sich einer der sechs Eingänge im Zustand logisch 1, ist der Ausgang des entsprechenden Inverters im Zustand logisch 0. Die übrigen Inverterausgänge sind dann zwangsläufig auch logisch 0, ebenfalls der Ausgang Q. Die Wahrheitstabelle für eine solche Schaltung ist in Tabelle 4-2 aufgeführt.

Die Schaltung funtioniert wie ein NOR-Gatter mit sechs Eingängen; in der Literaur findet man gelegentlich die Bezeichnung ''verdrahtetes NOR-Gatter''. In Bild 4-6 ist ein NOR-Gatter mit sechs Eingängen dargestellt. Die Schaltung in Bild 4-7 zeigt ein NAND-Gatter-Paar mit offenen Kollektorausgängen, die miteinander verdrahtet sind. Die Wahrheitstabelle für diese Schaltung finden Sie in Tabelle 4-3.

Tabelle 4-2. Wahrheitstabelle für die Schaltung in Bild 4-5

F	E	D	C	B	A	Q
0	0	0	0	0	0	1
alle anderen Zustände						0

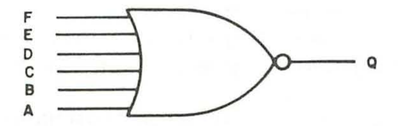


Bild 4-6. NOR-Gatter mit sechs Eingängen.

Tabelle 4-3. Wahrheitstabelle für die Schaltung in Bild 4-7. Die Schaltung wirkt als AND-OR-INVERTER mit zwei verschiedenen Eingängen.

D	C	B	A	Q
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

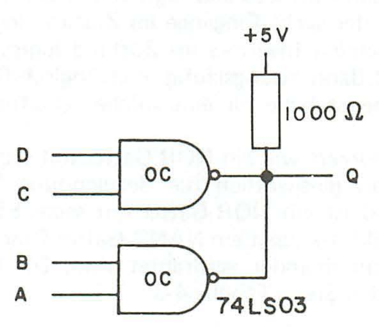


Bild 4-7. Zwei NAND-Gatter mit zwei Eingängen und miteinander verdrahteten, offenen Kollektorausgängen.

Wie aus der Tabelle 4-3 hervorgeht, ist Q = 0, wenn der Ausgang eines der Gatter im Zustand logisch 0 ist. Offene Kollektor-Gatter, die miteinander verdrahtet sind, verhalten sich recht ungewöhnlich. Sie können nicht als einfache Gatter benutzt werden! Es ist ratsam, mit aller Vorsicht ans Werk zu gehen, wenn Sie mehrere Gatter mit zwei Eingängen verwenden, deren Ausgänge miteinander verdrahtet sind.
Eine Ausnahme von dieser Regel bilden offene Kollektor-AND-Gatter, wie in der Schaltung Bild 4-8 gezeigt, deren Wahrheitstabelle in Tabelle 4-4 aufgeführt ist.
Das Symbol in der zweiten graphischen Darstellung sieht nicht wie ein AND-Gatter aus! Diese Art der Gatter-Darstellung wird in einem der folgenden Abschnitte dieses Kapitels erörtert.

Tabelle 4-4: Wahrheitstabelle für die Schaltung in Bild 4-8.

H	G	F	E	D	C	B	A	Q
1	1	1	1	1	1	1	1	1
alle anderen Zustände								0

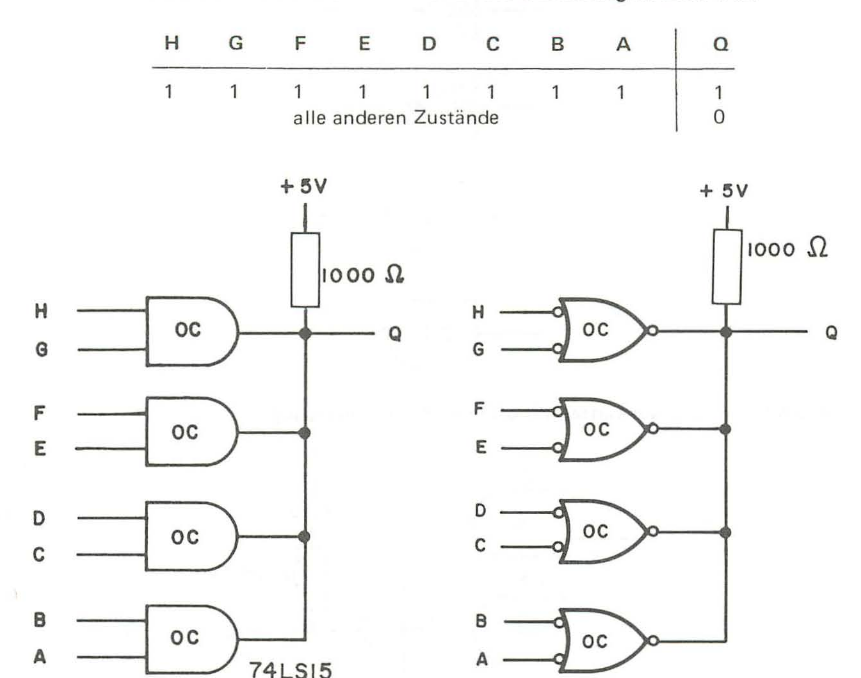


Bild 4-8. Zwei äquivalente Darstellungen von vier verdrahteten AND-Gattern mit je zwei Eingängen.

SYMBOLE FÜR GATTER MIT OFFENEN KOLLEKTORAUSGÄNGEN

Die Symbole (graphische Darstellungen) für Gatter mit offenen Kollektorausgängen sind leider nicht genormt. In den meisten Fällen werden herkömmliche Gattersymbole benutzt, und Sie können nur aus dem Text oder durch das Vorhandensein des ''Pull-Up-Widerstandes'' auf ein Gatter mit offenem Kollektor schließen. In diesem Buch werden die Buchstaben ''OC'' in den entsprechenden logischen Gattersymbolen benutzt, wie aus Bild 4-9 ersichtlich.

In verschiedenen Datenblättern sind die offenen Kollektorausgänge mit einem Stern (*) gekennzeichnet, wie Sie den Pinbelegungsskizzen für einige ICs mit offenem Kollektor in Bild 4-11 entnehmen können.

VERDRAHTETE OR-SCHALTUNGEN (WIRED-OR)

Schauen Sie sich die Schaltung in Bild 4-10 an.

Wenn sich entweder der Ausgang von Gatter 1 oder Gatter 2 oder Gatter 3 oder Gatter 4 im Zustand logisch 0 befindet, ist der Ausgang Q = logisch

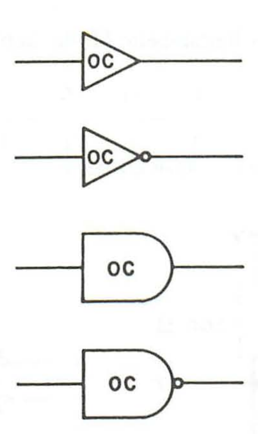


Bild 4-9. Symbole für Gatter mit offenem Kollektorausgang.

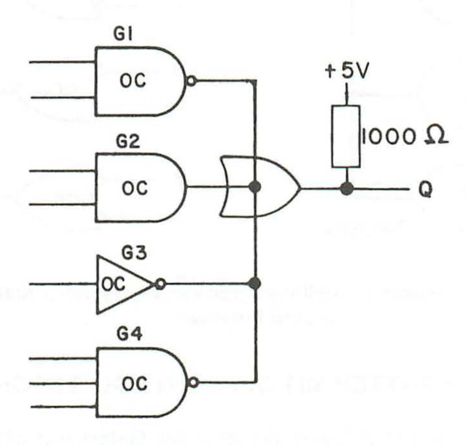
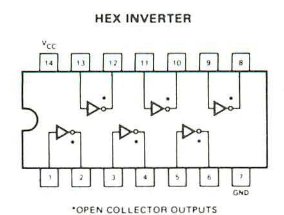


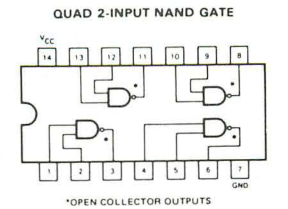
Bild 4-10. Verdrahtete OR-Schaltung.

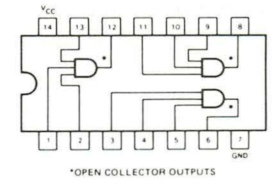
0. Diese Schaltung wird gelegentlich "verdrahtete OR-Schaltung" genannt, wie das OR-Gattersymbol mit Ausgang Q andeutet. Dieser Ausdruck kann jedoch irreführend sein, weil sich die Schaltung keineswegs wie ein einfaches OR-Gatter mit mehreren Eingängen verhält, was die Gattereingänge anbetrifft. Eine OR-Funktion ist schon eher an den Gattter*ausgängen* gegeben. Seien Sie daher vorsichtig, wenn Sie in der Eelektronik-Literatur auf den Ausdruck "verdrahtete OR-Schaltung" ("wired-OR") stoßen.

EINIGE TYPISCHE INTEGRIERTE SCHALTUNGEN MIT OFFENEM KOLLEKTOR

In Bild 4-11 einige der handelsüblichen ICs mit offenem Kollektor dargestellt.







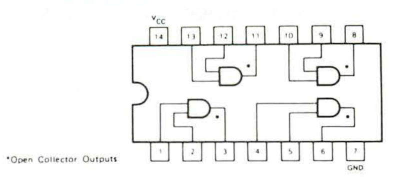


Bild 4-11. Einige integrierte Schaltungen mit offenem Kollektor.

POSITIVE GEGEN NEGATIVE LOGIK

Bei allen bisherigen Erörterungen wurde grundsätzlich folgendes festgestellt:

logisch 1 entspricht einem Potential von +5V
logisch 0 entspricht dem Massepotential

Dies nennt man *das positiv bewertete Logiksystem.* Ein AND-Gatter, dessen logisch bewertete Wahrheitstabelle in Tabelle 4-5 aufgeführt und dessen ELEKTRISCHES VERHALTEN in Tabelle 4-6 dargestellt ist, nennt man ein *positiv-logisches* AND-Gatter.

Tabelle 4-5. Logische Wahrheitstabelle für ein positiv-logisches AND-Gatter

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Tabelle 4-6. Elektrisches Verhalten eines positiv-logischen AND-Gatters (GND = Masse).

A	B	Q
GND	GND	GND
GND	+5V	GND
+5V	GND	GND
+5V	+5V	+5V

Was geschähe, wenn die vorstehenden Äquivalenz-Gleichungen umgekehrt würden und die physikalischen Gegebenheiten (z.B. bei AND- und OR-Gatter) gleich blieben? Dann würde die Äquivalenz wie folgt aussehen:

logisch 0 entspricht +5V
logisch 1 entspricht dem Massepotential

Dies nennt man *das negativ bewertete Logiksystem.* Eine *logische* Wahrheitstabelle auf der Basis von Tabelle 4-6 für ein *negativ-logisches AND-Gatter* ist in Tabelle 4-7 aufgeführt.

Tabelle 4-7. Wahrheitstabelle für ein negativ-logisches AND-Gatter

A	B	Q
1	1	1
1	0	1
0	1	1
0	0	0

Sie ist jedoch identisch mit der logischen Wahrheitstabelle für ein *positiv-logisches OR-Gatter.* Wenn Sie also mit Schaltungen zu tun haben, die dem negativ bewerteten logischen System entsprechen und für die Sie ein (negativ-logisches) AND-Gatter benötigen, ist das (positiv-logische) OR-Gatter-IC 74LS32 mit vier Gattern und zwei Eingängen die geeignete integrierte Schaltung. Das ist nicht so verwirrend, wie es sich anhört. Um das negativ logische Äquivalent eines gegebenen positiv-logischen ICs zu finden, brauchen Sie nur die Nullen und Einsen in der Wahrheitstabelle des positiv-logischen ICs gegeneinander auszutauschen und anhand des Ergebnisses das richtige Hardware-Gatter zu ermitteln. Beispiel: Welches ist das negativ-logische Äquivalent eines positiv-logischen NAND-Gatters mit zwei Eingängen? Um diese Frage zu beantworten, verfahre man wie folgt:

1. Schritt: Schreiben Sie die Wahrheitstabelle für ein positiv-logisches NAND-Gatter:

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

2. Schritt: Tauschen Sie die Nullen und Einsen gegeneinander aus und bilden eine negativ-logische Wahrheitstabelle (der Stern hinter dem Ausgang Q bedeutet, die Tabelle bezieht sich auf ein negativ-logisches IC).

A	B	Q*
1	1	0
1	0	0
0	1	0
0	0	1

3. Schritt: Versuchen Sie, das durch diese Tabelle definierte Gatter zu ermitteln; es ist das NOR-Gatter.

Folglich entspricht ein positiv-logisches NAND-Gatter einem negativ-logischen NOR-Gatter.

Nach demselben Verfahren finden Sie bestätigt, daß ein negativ-logisches NAND-Gatter einem positiv-logischen NOR-Gatter entspricht. Nachstehend einige weitere Äquivalenzen:

Ein positiv-logisches AND-Gatter mit zwei Eingängen entspricht einem negativ-logischen OR-Gatter mit zwei Eingängen.

Ein negativ-logisches AND-Gatter mit zwei Eingängen entspricht einem positiv-logischen OR-Gatter mit zwei Eingängen.

Ein positiv-logisches NAND-Gatter mit zwei Eingängen entspricht einem negativ-logischen NOR-Gatter mit zwei Eingängen.

Ein negativ-logisches NAND-Gatter mit zwei Eingängen entspricht einem positiv-logischen NOR-Gatter mit zwei Eingängen.

Es stellt sich die Frage nach dem Warum der negativen Logik. Erstens ist es verwirrend und zweitens werden die Hardware-Anforderungen bereits durch positiv-logische ICs erfüllt. Negative Logik ist jedoch oft hilfreich, wenn man komplizierte positiv-logische Schaltungen vereinfachen will. Das hängt hauptsächlich mit den offenen Kollektorausgängen zusammen.

Mit offenen Kollektorausgängen läßt sich eine logische 0 viel leichter aufspüren (oder elektrisch sichtbar machen). Wie bereits aus den Schaltungen in Bild 4-5 und Bild 4-8 hervorgeht, lassen sich Gatterausgänge mit offenem Kollektor zusammenschalten und mit dem BUS verbinden. Ist nur ein Gatterausgang logisch 0, überträgt sich dieses Potential auf den gemeinsamen BUS-Anschluß.

Bei digitalen Systemen ist jedoch logisch 1 und logisch 0 gleichbedeutend. Für positiv-logische Systeme gilt:

"Daten" = 1 und "keine Daten" = 0

Das heißt, die Leitung führt im Ruhezustand Massepotential und überträgt die digitale Information als +5V-Impulse. In Schaltungen mit offenem Kollektorausgang ist das Vorhandensein von logisch 0 bemerkenswert. Für die Datenübertragung gilt:

"Daten" = 0 und "keine Daten" = 1

Der Ruhezustand einer Leitung mit mehreren offenen Kollektorausgängen ist logisch 1. Das Zeitdiagramm in Bild 4-12 zeigt die Übertragung des 7-Bit-Digitalwortes 1010101 als positive und negative Signalfolge.

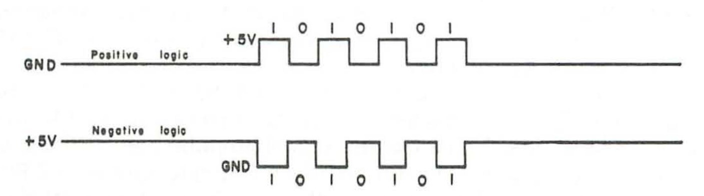


Bild 4-12. Positiv- und negativ-logisches Zeitdiagramm für die Aussendung des Wortes 1010101.

GEGENÜBERSTELLUNG VON POSITIVER UND NEGATIVER LOGIK

In den Schaltbildern sind Ihnen sicherlich kleine Kreise an den Ein- und Ausgängen der Gatter bei integrierten Schaltungen aufgefallen. Weitere Beispiele finden Sie in Bild 4-13. Diese kleinen Kreise nennt man *Umkehrkreise,* weil sie anstelle eines Inverters gezeichnet sind. Eine symbolische Darstellung der Äquivalenzen zwischen positiv-logischen und negativlogischen Gattern finden Sie in Tabelle 4-8.

Nun zur normalen Anwendung dieser Äquivalenzen beim Z-80-Interfacing! Die Mehrzahl der Z-80-CPU-*Steuer*signale sind low-aktiv und damit negativlogische Signale. Wenn Sie also einen *negativen* Impuls erzeugen wollen, während $\overline{\text{IORQ}}$ *und* $\overline{\text{RD}}$ aktiv sind, können Sie ein negativ-logisches AND-Gatter, wie in Bild 4-14 dargestellt, benutzen.

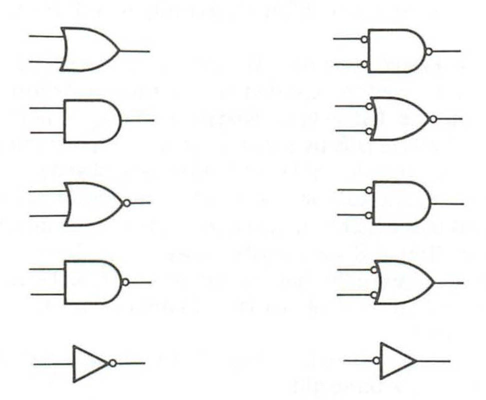


Bild 4-13. Beispiel für die Anwendung von Umkehrkreisen.

Dabei handelt es sich um ein positiv-logisches OR-Gatter, wie z.B. das IC 74LS32 mit vier Gatter und je 2 Eingängen. Wichtig bei der Konzipierung der Schaltung ist für Sie, zuerst die logische Funktion zu wählen, die Sie benötigen – in diesem Falle AND – und das Symbol zu zeichnen. Dann denken Sie an den Begriff *aktiv:* Wenn $\overline{\text{IORQ}}$ und $\overline{\text{RD}}$ *aktiv* sind, wird ein aktives Ausgangssignal benötigt! Falls *aktiv low* bedeutet, sind Umkehrkreise erforderlich. In diesem Falle bedeutet *aktiv* für beide Eingänge und für den Ausgang *low.* Welches Gatter wählen Sie, wenn die Aufgabe lautet: Erzeugen Sie nur dann einen positiven Impuls, wenn $\overline{\text{IORQ}}$ und $\overline{\text{RD}}$ aktiv sind? Da $\overline{\text{IORQ}}$ und $\overline{\text{RD}}$ bekanntlich *low*-aktiv sind und ein positiver Impuls high-aktiv, müßten Sie die in Bild 4-15 dargestellte Schaltung konzipieren.
Die Schaltung benötigt ein negativ-logisches NAND-Gatter oder ein positivlogisches NOR-Gatter. Geeignet ist also das Vierfach-NOR 74LS02. Die Schaltbilder in Kapitel 3 über Geräte-Auswahlimpulse bieten Ihnen weitere Beispiele für die Verwendung von negativen Gattersymbolen in Z-80-CPU-Interface-Schaltungen. Diese Gattersymbole finden auch im restlichen Teil des Buches Anwendung. Was Sie sich grundsätzlich merken müssen, ist

Tabelle 4-8. Äquivalenzen und Symbole für positiv- und negativ-logische Gatter

Funktion	Symbol der positiven Logik	Symbol der negativen Logik	Bemerkungen
OR			Ein positiv-logisches OR-Gatter entspricht einem negativ-logischen AND-Gatter.
AND			Ein positiv-logisches AND-Gatter entspricht einem negativ-logischen OR-Gatter.
NOR			Ein positiv-logisches NOR-Gatter entspricht einem negativ-logischen NAND-Gatter.
NAND			Ein positiv-logisches NAND-Gatter entspricht einem negativ-logischen NOR-Gatter.
INV			Ein positiv-logischer Inverter entspricht einem negativ-logischen Inverter.

Bild 4-14.

Bild 4-15.

folgendes: LOW-AKTIVE SIGNALE WERDEN ALS NEGATIV-LOGISCHE SIGNALE ANGESEHEN UND UMKEHRKREISE WERDEN ENTSPRECHEND BENUTZT: LOW-AKTIVE EINGÄNGE UND AUSGÄNGE HABEN ZUGEORDNETE UMKEHRKREISE.
Ähnliches gilt für Ein- und Ausgänge von Digitalgeräten und -Schaltungen.

Sie sind in Tabelle 4-9 zusammengefaßt. Auch hier gibt es wieder genügend Beispiele für die Verwendung von Umkehrkreisen in den Schaltbildern des 3. Kapitels. Die in Tabelle 4-9 aufgeführten Regeln bleiben auch für den Rest des Buches maßgebend.

GEGENÜBERSTELLUNG POSITIVER UND NEGATIVER BUS-LEITUNGEN

Ein weiterer Erörterungspunkt für positive und negative Logik ist die Unterscheidung zwischen einem *positiven* und einem *negativen* BUS. Bei einem positiven BUS ist der Ruhezustand = Masse, während er bei einem negativen BUS = +5V ist. Three-State-Logik führt zu positiven BUS-Systemen, während offene Kollektor zu negativen BUS-Systemen führt. Schaltbild 4-16 zeigt einen negativ logischen BUS, an dem vier Gatter

Tabelle 4-9. Potentielle Bedeutungen für Umkehrkreise an Ein- und Ausgängen von Digitalgeräten oder -Schaltungen

Symbole	Bemerkungen
Eingänge	Logisch 1 gibt das Gerät oder die Schaltung frei; logisch 0 blockiert das Gerät oder die Schaltung; durch positive Flanken werden Geräte oder Schaltungen freigegeben, abgetastet, getriggert oder getaktet; Daten werden nicht invertiert; der Eingangsruhezustand ist high; das Eingangssignal ist high-aktiv.
	Logisch 0 gibt das Gerät oder die Schaltung frei; logisch 1 blockiert das Gerät oder die Schaltung; durch negative Flanken werden Geräte oder Schaltungen freigegeben, abgetastet, getriggert oder getaktet; Daten werden invertiert; der Eingangsruhezustand ist high; das Eingangssignal ist low.
Ausgänge	Der Ausgang eines Gerätes oder einer Schaltung wird nicht invertiert; der Ausgang eines Gerätes oder einer Schaltung ist ein positiver Taktimpuls; der Ausgangsruhezustand ist low; das Ausgangssignal ist high-aktiv;
	Der Ausgang eines Gerätes oder einer Schaltung wird invertiert; der Ausgang eines Gerätes oder einer Schaltung ist ein negativer Taktimpuls; der Ausgangsruhezustand ist high; das Ausgangssignal ist low-aktiv.

mit offenem Kollektorausgang angeschlossen sind. Der Ruhezustand der BUS-Leitung liegt bei +5V. Geht ein Gatterausgang auf logisch 0 (oder Masse), wird die gesamte BUS-Leitung logisch 0. Erhält nun eines der Gatter G, F und E einen Freigabeimpuls, steht die BUS-Zustandsänderung am entsprechenden Gatterausgang zur weiteren Verarbeitung zur Verfügung.

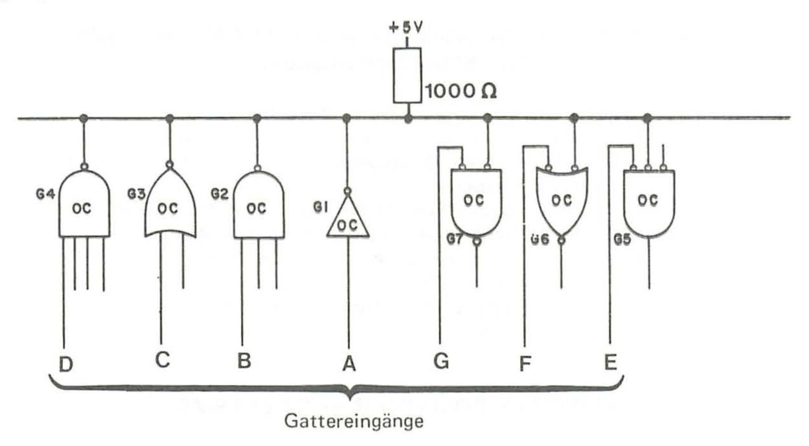


Bild 4-16. Negativ-logischer BUS.

Damit wird das Thema negative Logik und negative BUS-Leitungen beendet. Falls nichts anderes angegeben, sind in der Folge alle Logiken und BUS-Leitungen positiv.

PUFFER/SPERRSCHALTUNG

Die z.Zt. dominierende BUS-Technik, die bei den meisten Mikroprozessoren Anwendung findet, ist die positiv-logische Three-State-Technik. Eine typische Schaltungsversion, wie sie sowohl in Mikroprozessoren selbst als auch extern angetroffen wird, ist die Three-State-Puffer/Sperrschaltung, wie in Bild 4-17 dargestellt.

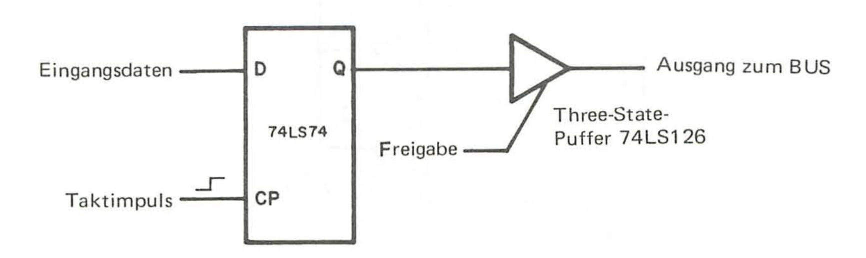


Bild 4-17. Three-State-Puffer/Sperrschaltung

Das Flipflop 74LS74 dient als Speicher, während der Three-State-Puffer die "Verbindung" des Flipflop-Ausgangs zum BUS steuert. Dazu gehört die Wahrheitstabelle 4-10. Mit dieser Art von Ausgangsschaltung ist es möglich, Ausgangsdaten in einer Sperre festzuhalten (gespeichert) und zu einem späteren Zeitpunkt auf den BUS zu legen. Das kann sehr wichtig sein, wenn viele Daten gleichzeitig auf den BUS wollen.

Tabelle 4-10. Wahrheitstabelle für die in Bild 4-17 dargestellte Three-State-Sperrschaltung

Taktgeber	Freigabe	Ausgabezustand
0	0	frühere Daten gespeichert; Three-State-Ausgang gesperrt;
0	1	gespeicherte Daten werden auf den BUS gelegt;
0-1-0	0	neue Dateneingabe; Three-State-Ausgang blockiert, daher keine neue Datenausgabe zum BUS;
0-1-0	1	neue Dateneingabe und sofortige Weitergabe zum BUS.

BEISPIELE POSITIVER BUS-SYSTEME

In Bild 4-18 wird ein einfaches, positives BUS-System mit vier Eingangssignalen und einer Leitung dargestellt, das auf der Verwendung eines einzigen Three-State-Puffer-ICs 74LS126 basiert. Diese Schaltung wird als ein BUS-System angesehen, weil die *Ausgänge der Gatter A bis D miteinander verbunden sind.* Mit den TTL-ICs der Standardserie 7400 ist dies nicht möglich, es sei denn, die ICs haben besondere Ausgangsschaltungen (entweder Three-State- oder offene Kollektorausgänge), die den Anschluß an einen BUS ermöglichen.

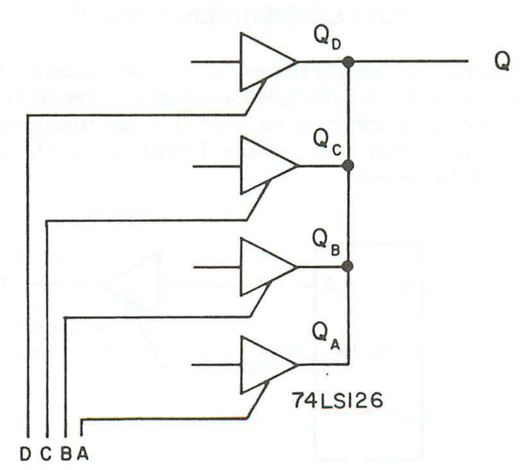


Bild 4-18. Positiv-logisches Vier-Geräte-BUS-System mit einer Leitung.

Wenn die Gatter A bis D in Bild 4-18 durch ein Signal logisch 1 freigegeben würden, müßte die Operation der Schaltung klar sein. *Es kann immer nur ein Puffer-Gatter freigegeben werden; die übrigen Puffer-Gatter müssen blockiert sein.* Daher erscheint zur selben Zeit digitale Information nur von einem der vier Puffer auf dem BUS mit einer Leitung. Die Information von den übrigen drei Puffern wird blockiert, da die entsprechenden Puffer gesperrt sind. Die Wahrheitstabelle 4-11 verdeutlicht die Operation der Schaltung.

Tabelle 4-11. Wahrheitstabelle für die Schaltung in Bild 4-18.

D	C	B	A	Ausgang
0	0	0	0	kein Gerät freigegeben
0	0	0	1	QA (Puffer B, C und D vom BUS getrennt)
0	0	1	0	QB (Puffer A, C und D vom BUS getrennt)
0	1	0	0	QC (Puffer A, B und D vom BUS getrennt)
1	0	0	0	QD (Puffer A, D und C vom BUS getrennt)

Wichtig ist folgende Feststellung: *Allen anderen logischen Bedingungen sind für diese Schaltung unzulässig, da sie Information von mehr als einem Puffer auf der einen Leitung des BUS erscheinen lassen. Falls Sie die unzulässigen Bedingungen zulassen, kann das Three-State-IC wahrscheinlich zerstört werden.*

In der Regel verfügen die BUS-Systeme anstatt über nur eine über mehrere Leitungen (Bild 4-19).

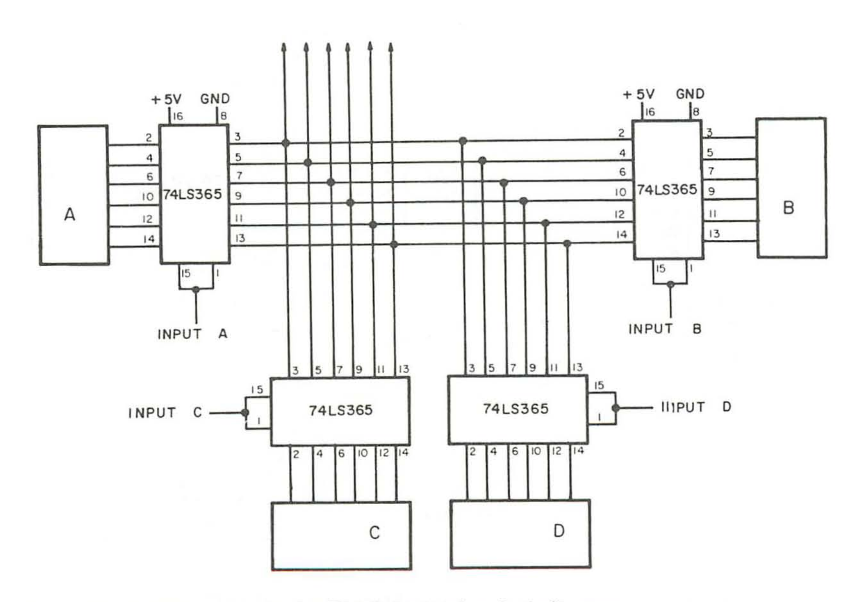


Bild 4-19. BUS-System mit vier Eingängen und sechs Leitungen.

Mit Ausnahme der Gattereingänge, die sechs Puffer gleichzeitig freigeben oder blockieren (aus diesem Grunde wird 74LS365 benutzt), stimmt diese Schaltung im Konzept mit der in Bild 4-18 überein. Die Eingänge A, B, C und D sind mit beiden Eingängen des externen NOR-Gatters im 74LS365 verbunden, das die sechs Freigabe/Blockier-Eingänge steuert, so daß bei logisch 0 die dazugehörigen 6-Bit-Ausgänge freigegeben werden. Die Wahrheitstabelle für diese Schaltung ist 4-12.

Tabelle 4-12. Wahrheitstabelle für die Schaltung in Bild 4-19.

D	C	B	A	Ausgang
1	1	1	1	Kein Gerät freigeben
1	1	1	0	Gerät A – sechs parallele Bits
1	1	0	1	Gerät B – sechs parallele Bits
1	0	1	1	Gerät C – sechs parallele Bits
0	1	1	1	Gerät D – sechs parallele Bits

DIE INNERE BUS-STRUKTUR DER Z-80-CPU

Zusätzlich zu den äußeren BUS-Leitungen (Adreß-, Steuer- und Daten-BUS) hat die Z-80-CPU einen inneren Three-State-Daten-BUS, der aus mindestens vier Funktionseinheiten und acht Leitungen besteht. Bild 4-20

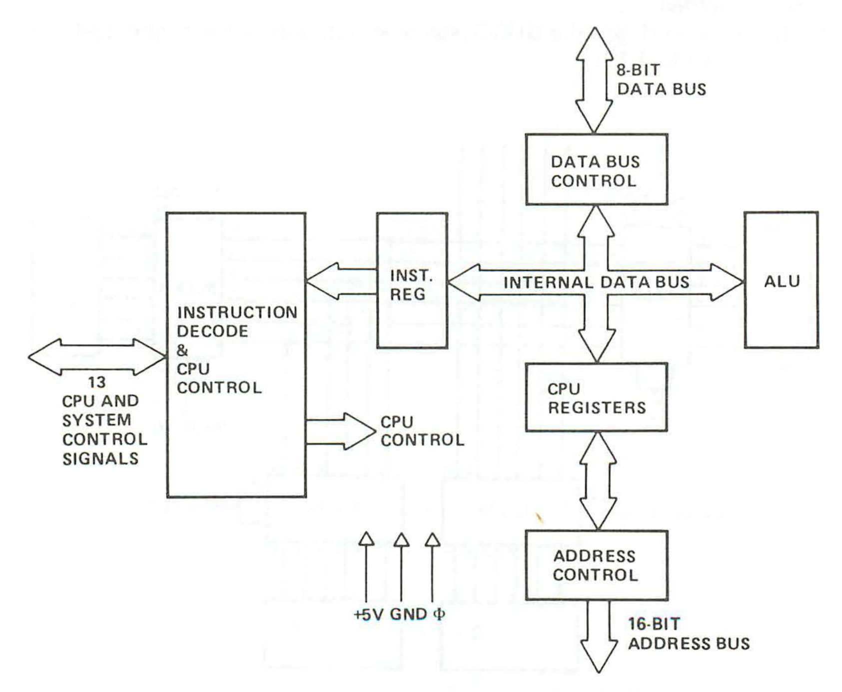


Bild 4-20. Innerer Daten-BUS und Funktionsblocks der Z-80-CPU.

zeigt den inneren Daten-BUS und sein Verhältnis zu den funktionellen Hauptelementen der Z-80-CPU und der externen BUS-Leitungen.
Die "Muttersprache" der Mikroprozessor-Beschreibungen ist Englisch. Deshalb findet man in der Literatur häufig die englischsprachigen Fachausdrücke. In diesem Buch ist dies bisher nicht der Fall. Ab jetzt jedoch bleiben die englischen Fachausdrücke – speziell in Blockschaltbildern – erhalten. Zur Verdeutlichung sich nachfolgend die Bezeichnungen aus Bild 4-20 übersetzt.

8-Bit Data BUS	– 8-Bit-Daten-BUS
Data BUS Control	– Daten-BUS-Steuerung
Arithmetic Logic Unit (ALU)	– Rechenwerk
Internal Data BUS	– interner Daten-BUS
Inst. Reg.	– Befehlsregister
Instruction Decode & CPU Control	– Befehlsdekoder & CPU-Steuerung
CPU System Control Signals	– CPU- und Systemsteuerungssignale
CPU Control	– CPU-Steuerung
CPU Registers	– CPU-Register
Adress Control	– Adressensteuerung
16-Bit Adress BUS	– 16-Bit-Adreß-BUS

Obwohl sich der innere Daten-BUS der Z-80-CPU in einer einzigen integrierten Schaltung befindet, weichen die Zugriffsregeln nicht von denen für die in diesem Kapitel bereits erörterten BUS-Leitungen ab.

MIKROCOMPUTER-EIN- UND AUSGANG IN EINER BUS-UMGEBUNG

Bild 4-21 zeigt einen BUS mit einer CPU, zwei Sendegeräten (TX1 und TX2) sowie zwei Empfangsgeräten (RX1 und RX2). Wenn TX1 Daten über die CPU an RX1 zu senden wünscht, geschieht dies in folgenden Schritten:

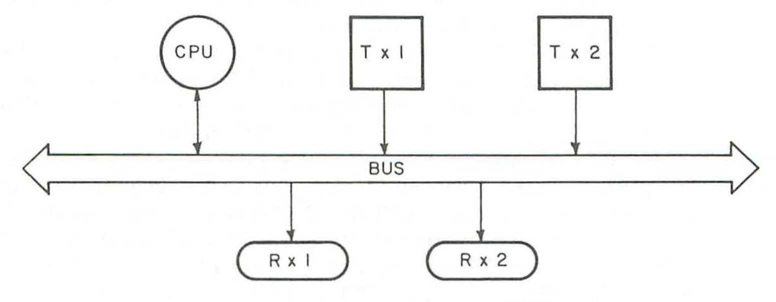


Bild 4-21. BUS mit einer CPU, zwei Sendern (TX1 und TX2) und zwei Empfängern (RX1 und RX2).

1. Die CPU empfängt Daten von TX1. (Somit ist ein Sender ein EINGABE-Gerät, das Daten *an* die CPU sendet.)

2. Die CPU sendet Daten an RX1. (Somit ist ein Empfänger ein AUSGABE-Gerät, das Daten *von* der CPU empfängt.)

Im restlichen Teil des Kapitels werden die Schritte 1 und 2 genauer untersucht; dabei soll TX1 ein 8-Bit-Puffer/Auffang-Register mit einem einzigen Daten-Byte und RX1 ein 8-Bit-Auffang-Register sein. Warum ist nicht RX1 ein Puffer/Auffang-Register? Die Beantwortung dieser interessanten Frage folgt zu einem späteren Zeitpunkt. Schauen Sie sich zunächst einen Mikrocomputer-Ausgang (also Schritt 2) an.

MIKROCOMPUTER-AUSGANG

Wenn die CPU für RX1 ein einziges Daten-Byte hat und ein Akkumulator ist, welche Hard- und Software ist notwendig, um das Byte über den BUS zu RX1 zu schicken? Wieviele Leitungen muß der BUS haben? Welche Signale muß der BUS übertragen? Wie weiß RX1, wann das Daten-Byte auf dem BUS ist? Nachstehend die Antworten.

Das vorige Kapitel hat die Befehle der I/O-Gruppe beschrieben, insbesondere den OUT A,(n)-Befehl. Er löst folgende Ereignisse auf dem Adreß-Daten- und Steuer-BUS aus:

1. Die CPU legt den Geräte-Code n auf die untere Hälfte (A0 . . . A7) des Adreß-BUS und den Inhalt des Akkumulators auf die obere Hälfte (A8 . . . A15) des Adreß-BUS.
2. Die CPU legt den Inhalt des Akkumulators auf den Daten-BUS.
3. Die CPU aktiviert gleichzeitig die Signale $\overline{IORQ}$ und $\overline{WR}$.
4. Die CPU deaktiviert die Signale $\overline{IORQ}$ und $\overline{WR}$.

Wie kann diese Ereignisfolge mit diesem speziellen Problem verbunden werden? Die zu den Ereignissen 1 bis 4 gehörenden Informationen lassen sich zu einem *Geräte-Auswahlimpuls* zusammenfassen, der für zwei wichtige Aufgaben zur Verfügung steht:

1. Durch Abtasten von RX1 anstelle von RX2 wählt der Geräte-Auswahlimpuls RX1 anstatt RX2 aus.
2. Der Geräte-Auswahlimpuls informiert RX1 über das Vorhanden sein von einem Datenbyte auf dem Daten-BUS.

Sobald die Akkumulator-Daten auf dem Daten-BUS zur Verfügung stehen, trifft der Geräte-Auswahlimpuls ein. Es gibt das 8-Bit-Auffang-Register bei RX1 frei, so daß es für den Datenempfang bereit ist. Unmittelbar nachdem sich die Akkumulaor-Daten auf dem Daten-BUS niedergelassen haben, taktet der Geräte-Auswahlimpuls das RX1-Flipflop. Die an den D-Eingängen anliegenden Daten gelangen zu den acht Q-Ausgängen. Man kann dies kontrollieren, indem man jeden der 8 Q-Ausgänge mit einem LED des LED-Monitors verbindet (siehe Versuch Nr. 5 am Ende dieses Kapitels). Folgende Signale für den Mikrocomputer-Ausgang müssen auf dem BUS erscheinen, der die CPU, RX1, RX2, TX1 und TX2 miteinander verbindet:

vom Daten-BUS: D0, D1, D2, D3, D4, D5, D6, D7
vom Adreß-BUS: A0, A1, A2, A3, A4, A5, A6, A7
vom Steuer-BUS: $\overline{IORQ}$, $\overline{WR}$

Wie Sie sicherlich festgestellt haben, wird die obere Hälfte des Adreß-BUS in diesem Fall nicht benötigt, weil der Geräte-Code nur auf der

unteren Hälfte erscheint. Ist jedoch noch ein Speicher mit angesprochen, sind alle 16 Adressenleitungen erforderlich. Damit das System flexibel bleibt, sind folgende Signale hinzugefügt:

Vom Adreß-BUS: A8, A9, A10, A11, A12, A13, A14, A15

Vom Steuer-BUS: $\overline{MREQ}$, $\overline{RD}$

Mit diesen Signalen und einer entsprechenden Dekoderschaltung können Ein- und Auslesevorgänge an beiden I/O-Geräten und eventuellen Speicherstellen durchgeführt werden. Genau genommen sind nicht alle Geräte physikalisch mit allen BUS-Leitungen verbunden. Z.B. werden RX1 und die dazugehörige Geräte-Auswahlschaltung nur an die zuerst aufgeführte Signalgruppe angeschlossen. Die gleiche Geräte-Auswahlschaltung ließe sich auch zum Abtasten von RX1 und RX2 oder sogar aller vier Geräte, RX1, RX2, TX1 und TX2 verwenden.

Die Schaltung zur Erzeugung der Geräte-Auswahlimpulse benutzt ein 1-zu-4-Dekoder/Demultiplexer-Dual-IC 74LS155. Das IC wird über eine "Wired-OR-Schaltung freigegeben, deren Eingangssignale auf den Adreßleitungen A2 . . . A7 sind. Die Freigabe des 74LS155 erfolgt nur dann, wenn A2 . . . A5 logisch 0 und A6/A7 logisch 1 sind. A0 und A1 wählen einen der vier low-aktiven Ausgänge der beiden unabhängigen Dekoder/Demultiplexer. Die Aktivierung des Dekoder/Demultiplexers wird von den

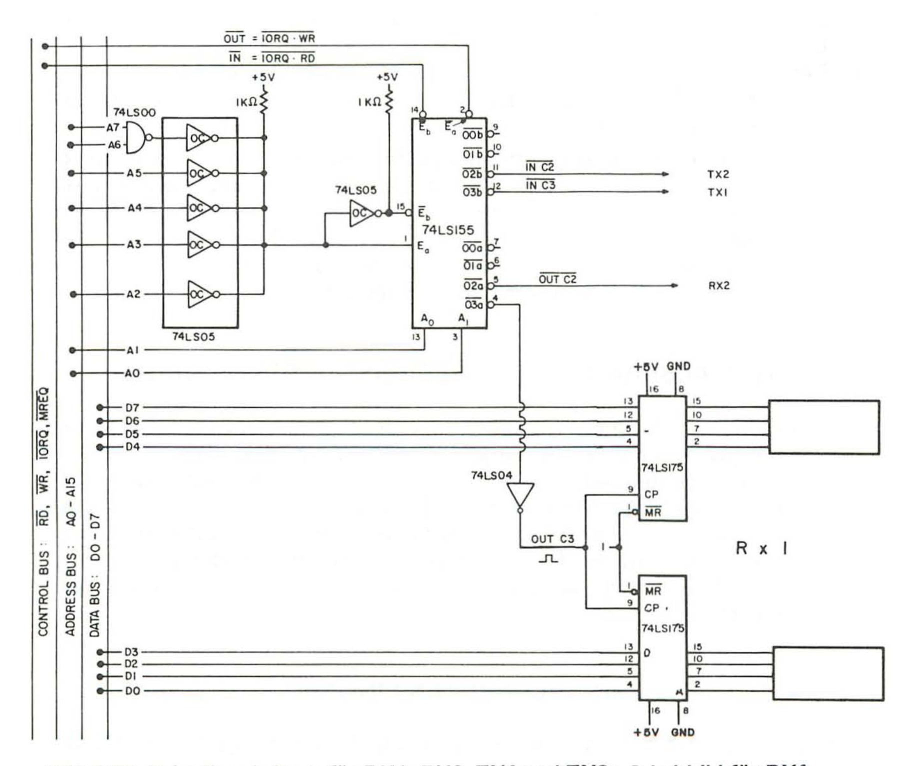


Bild 4-22. Dekodierschaltung für RX1, RX2, TX1 und TX2; Schaltbild für RX1.

$\overline{E_a}$- und $\overline{E_b}$-Eingängen an den Anschlußstiften 2 und 14 gesteuert. Wenn sowohl $\overline{IORQ}$ und $\overline{WR}$ low sind, erhält $\overline{E_a}$ ein Eingangssignal, daß den "a"-Dekoder/Demultiplexer aktiviert. Wie aus dem Schaltbild 4-22 ersichtlich, ist er für die Ausgangssignale zuständig und tastet die Empfänger RX1 und RX2 ab. Der $\overline{E_b}$-Eingang erhält den Eingangsimpuls, sobald die Signale $\overline{IORQ}$ und $\overline{RD}$ auf logisch 0 gehen. Dadurch wird der "b"-Dekoder/Demultiplexer aktiv und tastet die Sender TX1 und TX2 ab. Die Dekodierung des Geräte-Codes auf den 8 niederwertigen Adreß-BUS-Leitungen ist *absolut*. Das heißt, es ist nur eine aktivierte Ausgangsleitung frei. Bei der Schaltung in Bild 4-22 handelt es sich im Prinzip um ein bekanntes Konzept. Neu sind lediglich der BUS-orientierte Adreßbezugsweg, die Steuerleitungen von der CPU und die zwei 4-Bit-Auffang-Register (RX1). In der Schaltung ist sowohl der Adreß-, der Daten- als auch der Steuer-BUS vorhanden. Diese "Super-BUS-Leitung" nennt man in der Regel *System-BUS*. Kapitel 5 macht Sie mit dem System-BUS des Nanocomputers® bekannt.

Die zur Ausgabe von Daten aus einem Register an ein Ausgabegerät angewandte Technik läßt sich wie folgt zusammenfassen:

a) Erzeugung eines einfachen Ausgabegeräte-Auswahlimpulses über Software. Die gewöhnlich dazu benutzten Befehle sind OUT (n),A und OUT (C),r, wobei r jedes 8-Bit-Allzweckregister, A, B, C, D, E, H oder L sein kann.
b) Benutzung des Geräte-Auswahlimpulses zur Freigabe eines Auffang-Register-ICs in dem Augenblick, wo die Registerdaten auf dem bidirektionalen Daten-BUS erscheinen.

Die Z-80-CPU ist für den gesamten Synchronisationsvorgang verantwortlich. Die Auffang-Register-ICs spielen bei der Datenübertragung eine passive Rolle und fangen die Daten nur dann auf, wenn Sie durch einen Geräte-Auswahlimpuls dazu aufgefordert werden. Zum Auffangen der Daten wird je nach der Art des verwandten Auffang-Register-ICs ein positiver oder negativer Geräte-Auswahlimpuls benutzt.

EINIGE AUSGANGS-AUFFANG-REGISTER-SCHALTUNGEN

In den Bildern 4-22 und 4-23 sind einige der häufig benutzten Ausgangs-Auffang-Register-Schaltungen dargestellt. Um die Zeichnungen zu vereinfachen und zu verdeutlichen, wie die Auffang-Register-ICs aufgebaut sind, fehlt in Bild 4-23 die Schaltung zur Erzeugung des Geräte-Auswahlimpulses, der die Auffang-Register-Schaltungen abtastet. Stattdessen wird auf frühere Schaltbilder für Geräte-Auswahlimpulserzeugung verwiesen und in Bild 4-23 gezeigt, wo die Verbindung für die Auswahlimpulsschaltung herzustellen ist.

MIKROCOMPUTER-EINGANG

Betrachten Sie noch einmal den BUS in Bild 4-21 mit zwei Sendern, zwei Empfängern und einer CPU. Zwei Schritte sind erforderlich, um ein ein-

+5V GND
D7
D6
SDI
SDO
CDI
CDO
CPO
CPI
74LS74
OUT CIH
74LS174
CP
MR
D5
D4
D3
D2
D1
D0
LM7
LM6
LM5
LM4
LM3
LM2
LM1
LM0
LED-
Monitor
+5V GND

Bild 4-23. Auffang-Register-Schaltungen eines Mikrocomputers mit den Flipflops 74LS74 und 74LS174.

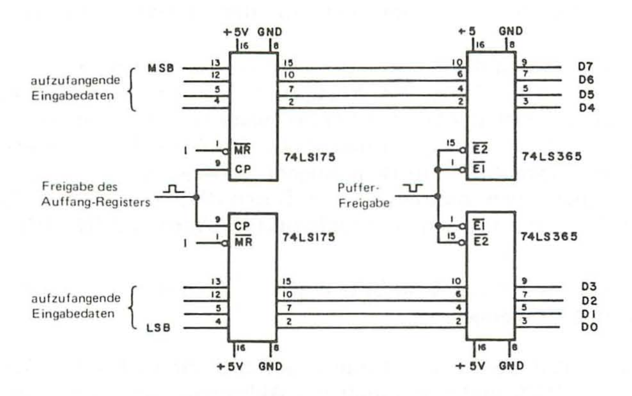


Bild 4-24. Eingangs-Puffer/Auffang-Register-Schaltung zwei Flipflops 74LS175 mit vier Puffern sowie zwei 74LS365 mit sechs Puffern.

ziges Daten-Byte über die CPU nach RX1 zu senden:
1. Die CPU empfängt Daten von TX1;
2. Die CPU sendet Daten nach RX1.

Während in dem Abschnitt über den Mikrocomputer-Ausgang Schritt Nr. 2 beschrieben ist, befaßt sich dieser Abschnitt über den Mikrocomputer-Eingang mit Schritt Nr. 1.
TX1 ist ein Puffer/Auffang-Register. Die zum Register gehörenden Flipflops können ein Datenbyte speichern. Die Q-Ausgänge der Flipflops sind durch blockierte Puffer vom BUS logisch getrennt. Die entsprechende Schaltung zeigt Bild 4-24.
Die Puffer/Auffang-Register-Schaltung ermöglicht dem Sendegerät TX1, Daten zu halten, ohne sie auf den BUS zu legen. Ohne die vorhandenen Puffer wären die Q-Ausgänge des Flipflops direkt mit dem BUS verbunden und die in TX1 enthaltenen Daten stünden dort jederzeit zur Verfügung. Wie bereits erwähnt, ist jedoch nur immer eine Datenübertragung auf dem BUS möglich, da ansonsten die Information verfälscht wird. Damit ist auch die Frage beantwortet, warum bei TX1 das 8-Bit-Auffangregister eine Pufferstufe haben muß. Ohne sie würden die Daten aus TX1 sofort auf den BUS übertragen und somit jede weitere Kommunikation blockiert.
Diese Situation darf unter keinen Umständen eintreten. Normalerweise verfügen Sendegeräte über ein Auffang-Register, das Daten über einen längeren Zeitraum speichern kann, währenddessen die Kommunikation zwischen anderen Geräten auf dem BUS stattfindet. Mit Hilfe des Enable/Disenable (Freigabe/Blockade) der Pufferstufe kann man die Zeit, in der sich die Daten auf dem BUS befinden, auf eine relativ kurze Periode beschränken. Die Daten sollen dann auf dem BUS sein, wenn das Empfangsgerät zur Aufnahme bereit ist. Die Auffang-Register in einem Sender ermöglichen also diesem, die zu sendenden Daten solange zu speichern, bis die eigentliche Übertragung beginnt. Die Perioden der Datenspeicherung und der Datenübertragung sind unabhängig voneinder.
Da die Q-Ausgänge der Auffang-Register in den Empfangsgeräten nicht mit dem BUS verbunden sind, sind keine Pufferstufen erforderlich. In dieser Hinsicht gehören die Empfänger zu den verhältnismäßig einfachen Geräten.
Die zur Ausführung des 1. Schrittes erforderliche Technik, nämlich Daten von TX1 in ein CPU-Register einzugeben, ist dem beim Mikrocomputer-Ausgang geschilderten analog. Über Software und eine Hardware-Dekoderschaltung wird ein einziger Eingangs-Geräte-Auswahlimpuls erzeugt, um einen Three-State-Puffer in dem Augenblick freizugeben, wo ein direkter Pfad zwischen dem bidirektionalen Daten-BUS und dem CPU-Register offen ist. Die dafür typischen Befehle sind IN A,(n) und IN r,(C).

Insbesondere, wenn die Z-80-CPU einen IN A,(n)-Befehl ausführt, wickeln sich folgende Vorgänge ab:

1. Die CPU legt den Geräte-Code n auf die untere Hälfte (A0 . . . A7) des Adreß-BUS und den Inhalt des Akkumulators auf die obere Hälfte (A8 . . . A15).
2. Die CPU aktiviert sowohl das Signal $\overline{IORQ}$ als auch $\overline{RD}$.

3. Die CPU liest den Inhalt des Daten-BUS (D0 . . . D7) in den Akkumulator.
4. Die CPU deaktiviert die Signale $\overline{IORQ}$ und $\overline{RD}$.

Um Eingabedaten erfolgreich in ein CPU-Register einzulesen, können die zu Vorgang 1 und 2 gehörigen Informationen durch Hardware-Dekodierung zur Erzeugung eines Geräte-Auswahlimpulses zusammengefaßt werden. Dieser Geräte-Auswahlimpuls wird für zwei wichtige Aufgaben benutzt:
1. Der Geräte-Auswahlimpuls wählt TX1 anstelle von TX2 als Sender durch Abtasten von TX1 anstatt TX2.
2. Der Geräte-Auswahlimpuls meldet TX1 die Bereitschaft der CPU, die Daten vom Daten-BUS abzulesen.

Der Geräte-Auswahlimpuls ist aktiv, unmittelbar bevor die CPU den Daten-BUS abliest. Man kann den Geräte-Auswahlimpuls dazu benutzen, die Puffer in der Puffer/Auffang-Register-Schaltung freizugeben, wodurch die Ausgänge der acht Flipflops für kurze Zeit direkt mit dem Daten-BUS verbunden sind.

Nachstehend eine Aufstellung sämtlicher für den Mikrocomputer-Eingang benötigten Signale, die auf dem Daten-BUS erscheinen müssen, um CPU, TX1, TX2, RX1 und RX2 miteinander zu verbinden:

vom Daten-BUS: D0, D1, D2, D3, D4, D5, D6, D7;
vom Adreß-BUS: A0, A1, A2, A3, A4, A5, A6, A7;
vom Steuer-BUS: $\overline{IORQ}$, $\overline{RD}$.

Wie bei Ausgabeoperationen ist die CPU auch bei Eingabeoperationen für den gesamten Synchronisationsvorgang verantwortlich. Der Three-State-Puffer spielt bei dem Datenübertragungsvorgang eine passive Rolle und legt nur dann Daten auf den Daten-BUS, wenn ihn ein Geräte-Auswahlimpuls dazu auffordert.
Je nach Art des benutzten Puffers wird entweder ein negativer oder ein positiver Geräte-Auswahlimpuls zur Freigabe des Puffers benutzt.
Es darf bekanntlich nur ein Three-State-Puffereingang zu einer Z-80-CPU freigegeben werden! Alle Eingangs-Geräte-Auswahlimpulse müssen *absolut dekodiert* sein, was für das I/O-Gatter bedeutet: alle acht Bits des Geräte-Codes müssen benutzt werden, um das gewünschte Eingabegerät zu dekodieren. Erhält ein nichtexistentes Gerät die Aufforderung Daten einzugeben, liest das entsprechende Register in der Regel das Byte FF ein.

Nachstehend eine Zusammenfassung der Grundtechnik für die Dateneingabe in ein CPU-Register:
a) Erzeugung eines einzigen Eingangs-Geräte-Auswahlimpulses über Software. Die dazu benutzten Befehle sind IN A,(n) und IN r,(C), wobei r jedes 8-Bit-Allzweckregister, A, B, C, D, E, H oder L sein kann.
b) Benutzung des Geräte-Auswahlimpulses, um die Three-State-Puffer im Sendegerät freizugeben. Sobald die Puffer freigegeben sind, können die zu sendenden Daten auf den Daten-BUS gelegt werden, woraufhin die CPU die Daten in eins der internen Register einliest; in welches Register hängt von der Art des ausgeführten Eingebebefehls ab.

Bei Mikrocomputer-Eingabeoperationen steuert die CPU den Auffangvorgang von Daten in die Sende-Flipflops oft direkt. Bild 4-24 zeigt eine Schaltung, bei der ein Geräte-Auswahlimpuls benutzt wird, um Daten in die Sende-Flipflops zu takten, während ein zweiter Geräte-Auswahlimpuls die Puffer zwischen den Q-Ausgängen des Sende-Flipflops und dem Daten-BUS freigibt.

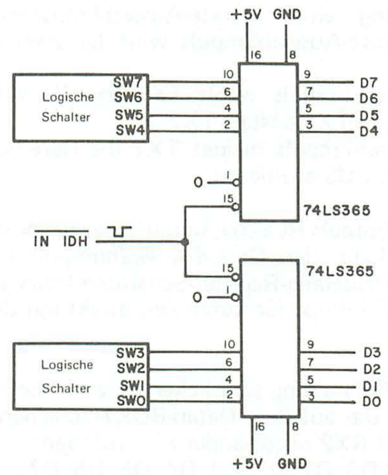


Bild 4-25. Eingabe-Schaltung mit acht logischen Schaltern und dem IC 74LS365 mit sechs Puffern.

EINIGE PUFFER/AUFFANG-REGISTER-EINGABE-SCHALTUNGEN

Bilder 4-24 und 4-25 zeigen Schaltbilder von zwei Puffer/Auffang-Eingabe-Schaltungen. In Bild 4-25 führen die logischen Schalter dieselbe Funktion aus wie ein Auffang-Register, nämlich das Festhalten (Speichern) von Daten. Die ICs 74LS365 haben je sechs Puffer mit einer normalen Freigabe/Blockierungsleitung. Die Eingänge dieser Puffer sind mit den Ausgängen der logischen Schalter verbunden, während die Pufferausgänge an dem Daten-BUS D0 . . . D7 angeschlossen sind. Über einen Geräte-Auswahlimpuls werden die Puffer freigegeben und legen so die Eingabe (oder Senderdaten) auf den Daten-BUS. Die logischen Schalter und die Puffer zusammen bilden ein Beispiel dafür, wie Hardware den Sender TX1 in Bild 4-21 realisiert.

Die Schaltung in Bild 4-24 zeigt zwei Auffang-Register 74LS175 mit vier Puffern und zwei Auffang-Register 74LS365 mit sechs Puffern als weiteres Hardware-Beispiel für TX1. Die Eingänge zu den Auffang-Registern 74LS175 können an irgendeinem Eingang des TX1 angeschlossen werden. Die Eingänge der Auffang-Register 74LS175 kann man an jede digitale Datenquelle (z.B. Laborgeräte) anschließen. Die Dateneingabe in ein CPU-Register erfolgt in zwei Schritten:

1. Auffangen der Daten in den Flipflops 74LS175;
2. Erzeugung eines Geräte-Auswahlimpulses, um die Puffer 74LS365 frei-

zugeben, damit die Daten auf den Daten-BUS gelegt und ins Register eingelesen werden können.

Die Versuche erfolgen mit ähnlichen Schaltungen wie in diesem Kapitel beschrieben.

I/O-TECHNIKEN MIT SPEICHERPLAN

In den vorhergehenden Abschnitten dieses Kapitels sind I/O-Operationen beschrieben, die von Befehlen der I/O-Befehlsgruppe, z.B. IN A,(n), IN r,(C), OUT n,(A) und OUT (C),r ausgelöst werden. Die Art der Mikrocomputer-Ein- und Ausgaben nennt man GATTER EIN- AUSGABEN, weil daran eine Anzahl von 8-Bit-Ein- Ausgabe-Gatter beteiligt sind. Die Software bestimmt, welche Operation auszuführen ist. Die zu der Gatter-I/O gehörenden Signale sind $\overline{IORQ}$ und $\overline{RD}$ für die Eingabe und $\overline{IORQ}$ bzw. $\overline{WR}$ für die Ausgabe.
Es gibt noch eine weitere Methode der Mikrocomputer-I/O; sie nennt sich *I/O mit Speicherplan* und ist Gegenstand dieses Abschnitts. Bei einer I/O-Operation mit Speicherplan wird ein Eingabe-/Ausgabegerät im wesentlichen so behandelt wie eine Speicherstelle. Somit hat das Gerät eine 16-Bit-Adresse und erhält Zugriff durch den Befehl, der Speicherzugriffe auslöst, wie z.B. LD-Befehle. Um z.B. ein Informations-Byte vom B-Register zu einem Gerät an der Speicherstelle hex C325 mit der Speicherplan-I/O-Technik auszugeben, könnte man den Befehl LD (0C325H),B benutzen. Ähnlich wäre ein Speicherplan-Eingabefehl vom Gerät ins Register B = LD B, (0C325H).

Im Grunde genommen sind Speicherplan-I/O und Gatter-I/O sehr ähnlich. In jedem Falle zeigt ein Steuersignal ($\overline{RD}$ oder $\overline{WR}$) an, ob es sich bei der Operation um eine Ein- oder Ausgabe handelt. Ebenfalls muß der Adreß-BUS (nur A0 . . . A7 für Gatter-I/O) dekodiert werden, um ein spezielles I/O-Gerät zu erkennen. Letztlich findet die eigentliche Datenübertragung immer während eines Maschinenzyklus bei der Ausführung eines Befehls statt. Eine Speicherplan-I/O ist genauso wie der Zugriff auf eine Speicherstelle; nur handelt es sich bei der adressierten Schaltung nicht um Speicher-ICs, sondern um eine Art I/O-Gerät.
Eine Speicherplan-Eingabeoperation benötigt folgende Schritte:
1. die CPU legt die 16-Bit-Geräteadresse auf den Adreß-BUS;
2. die CPU aktiviert die Signale $\overline{MREQ}$ und $\overline{RD}$;
3. die Information der beiden ersten Schritte wird in die Dekoderschaltung eingegeben. Sie erzeugt einen Impuls, der die Puffer im Eingabegerät freigibt, wodurch die Eingabedaten zum Daten-BUS gelangen;
4. die CPU liest die Daten vom Daten-BUS ab;
5. die CPU deaktiviert die Signale $\overline{MREQ}$ und $\overline{RD}$, infolgedessen auch den Auswahlimpuls, so daß die Puffer im Eingabegerät sperren und das Eingabegerät zu einem hochohmigen Zustand zurückkehrt;
6. der BUS steht für weitere Kommunikationen zur Verfügung.

Die Speicherplan-Ausgabe hat eine ähnliche Schrittfolge.

1. Die CPU legt die Ausgabedaten auf den Daten-BUS und die 16-Bit-Geräteadresse auf den Adreß-BUS.
2. Die CPU aktiviert die Signale $\overline{MREQ}$ und $\overline{WR}$.
3. Die Information der Schritte 1 und 2 wird in die Dekoderschaltung eingegeben, die einen Impuls erzeugt, der die Flipflops im Ausgabegerät taktet und die Ausgabedaten auffängt.
4. Die CPU deaktiviert die Signale $\overline{MREQ}$ und $\overline{WR}$, nimmt die Geräteadresse vom Geräte-BUS und die Daten vom Daten-BUS, indem sie einige interne Puffer in einen hochohmigen Zustand zurückführt.
5. Weitere Kommunikationen auf dem BUS können wiederaufgenommen werden.

Nur wenige Vorteile sprechen für die Benutzung einer Speicherplan-I/O anstelle einer Gatter-I/O. Während die 16-Bit-Adresse die Dekoderschaltungen komplexer macht, dehnt sie den Raum der Geräteadresse über die bei der Gatter-I/O vorhandenen 256 aus. Für Ausgabegeräte, die summieren oder zählen, ist eine Erhöhung mit Hilfe eines Befehls INC (HL) schneller möglich als mit der dreifachen Befehlsfolge (IN A,(n), INC A, OUT (n),A. Sie wäre erforderlich, wenn man eine Gatter-I/O benutzt (vorausgesetzt, die Adresse ist bereits im HL gespeichert). Wenn die Adresse im HL-Registerpaar gespeichert ist, sind Befehle, wie LD r(HL) und LD (HL),r schneller als ihre Gegenstücke IN/OUT.
Die Methode zur Erzeugung von Speicherplan-Synchronisationsimpulsen ist mit der zur Erzeugung von Speicherzugriff-Synchronisationsimpulsen identisch. Auch das Verdrahten von Eingangs-Puffer/Auffang-Register und Ausgangs-Auffang-Register für Speicherplan-I/O ist mit den Schaltungen für Gatter-I/O identisch.
In Kapitel 3 sowie in diesem Kapitel sind entsprechende Schaltbilder vorhanden.

MIKROCOMPUTER-BUS-LEITUNGEN

Wie bereits in Kapitel 2 erwähnt, werden die CPU-Anschlüsse in den meisten Mikrocomputersystemen gepuffert, um einen besseren Schutz und eine höhere Treibfähigkeit zu gewährleisten. Neben der Pufferung sind die CPU-Signale oft zu neuen Signalen zusammengefaßt, wie z.B. $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IN}$ und $\overline{OUT}$. Für bestimmte Standard-BUS-Leitungen (z.B. den S-100-BUS, den INTEL Multibus® und den IEEE-488-BUS) die nicht speziell für Z-80-CPUs konzipiert sind, ist zusätzliche Logik erforderlich, um den BUS-Standard-Spezifikationen zu entsprechen. Der Nanocomputer®-BUS ist dazu bestimmt, viele CPUs zu unterstuützen. Darum sind bestimmte Signale erforderlich, um zu einer gegebenen CPU-Adresse, zu Daten- und Steuer-BUS-Leitungen externen CPU-Zugriff zu finden. Folgende Feststellung ist wichtig:
CPU-Signale erscheinen selten auf einem Mikrocomputer-BUS.
Im Falle des Nanocomputers®, der eine neue Norm benutzt – den "Gamma-BUS" – sind die meisten CPU-Signale durch einen einzigen Puffer torgesteuert. Die Signale bleiben also unverändert, nur um max. 30 Nanosekunden verzögert.

Nachstehend die BUS-Signale, bei denen es sich lediglich um CPU-Signale handelt, die durch einen einzigen Puffer torgesteuert (und in den meisten Fällen freigegeben) sind:

$$BA0 \ldots BA15, \overline{BWR}, \overline{BRD}, \overline{BIORQ}, \overline{BMREQ}, \overline{BBUSAK},$$
$$\overline{BHALT}, \overline{BMI}, \overline{BRFSH}, \text{und } B\Phi$$

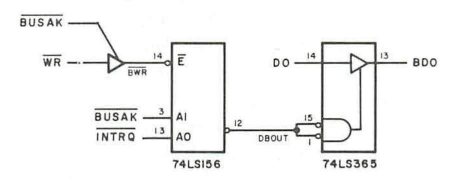


Bild 4-26. Schaltung zur Freigabe von CPU-Daten für den Nanocomputer®-BUS.

Bild 4-27. Zeitdiagramm für eine I/O-Schreiboperation zum Nanocomputer®-BUS.

CPU-Signale, die vor dem Anschluß am BUS gepuffert sein sollen, sind:

$\overline{BWAIT}$, $\overline{BINT}$, $\overline{BNMI}$, $\overline{BBUSRQ}$ und $\overline{BRESET}$.

Alle diese Eingangssignale sind durch einen 930-Ohm-Widerstand mit +5 V verbunden. Die Datenleitungen DB0 ... BD7 sind in den obigen Signalaufzählungen nicht enthalten. Die Puffer, welche D0 ... D7 zum Nanocomputer®-BUS steuern, werden durch das Signal $\overline{DBOUT}$ für den Ausgang und durch DBIN für den Eingang freigegeben. Denken Sie daran, der Daten-BUS ist bidirektional! Bild 4-26 verdeutlicht, wie das $\overline{DBOUT}$-Signal zur Freigabe der CPU-Daten auf Pin D0 für die entsprechenden BUS-Leitung BD0 des Nanocomputers® erzeugt wird.
Analysieren Sie nun das Zeitdiagramm einer I/O-Schreiboperation mit maximaler Verzögerung; die Ereignisse sind wie folgt:

1. Die CPU aktiviert $\overline{IORQ}$. Kurz darauf legt die CPU Daten auf D0 . . . D7.
2. Die CPU aktiviert $\overline{WR}$. Der genaue Zeitintervall nach der Aktivierung von $\overline{IORQ}$ ist nicht wichtig.
3. $\overline{WR}$ wird durch einen von $\overline{BUSAK}$ freigegebenen Puffer gesteuert, um das $\overline{BWR}$-Signal zu erzeugen. Die Verzögerung beträgt 16 ns (max.).
4. $\overline{BWR}$ gibt den Dekoder 74LS156 frei, um ein low-aktives $\overline{DBOUT}$-Signal zu erzeugen. Die Verzögerung beträgt 30 ns (max.).
5. Das $\overline{DBOUT}$-Signal gibt einen Puffer 74LS365 frei, der die Daten der Leitung D0 zur BUS-Leitung BD0 weiterleitet. In ähnlicher Weise sind die anderen Datenleitungen am BUS angeschlossen. Die Verzögerung beträgt 30 ns (max.).

Diese Ereignisse sind im Zeitdiagramm 4-27 dargestellt.
Angenommen, Sie möchten eine Ausgangs-Auffang-Register-Schaltung für einen Mikrocomputer mit Hilfe von $\overline{WR}$ und $\overline{IORQ}$ verdrahten, um das Auffang-Register zu takten. Wie in Bild 4-27 deutlich zu sehen, sind die Daten nicht auf dem Daten-BUS BD, wenn $\overline{WR}$ aktiv wird. Wie wäre es mit der Benutzung von $\overline{BWR}$ anstelle von $\overline{WR}$? Selbst beim aktivierten $\overline{BWR}$-Signal stehen die Daten noch nicht zur Verfügung. Das $\overline{BWR}$-Signal muß noch weiter verzögert werden. Erst wenn sich das $\overline{WR}$-Signal in das verzögerte und gepufferte $\overline{DBWR}$-Signal umgewandelt hat, stehen die Daten auf den Leitungen BD0 . . .BD7 zum Abruf bereit. Hieraus wird deutlich, warum für Ausgangsschaltungen des Nanocomputers® die Benutzung von $\overline{DBWR}$ wichtig und notwendig ist.

Die nachfolgenden Versuche machen Sie mit den offenen Kollektorausgängen, der Three-State-BUS-Technik sowie mit einigen Schaltungen für Mikrocomputer-I/O vertraut.

Versuch-Nr.	Bemerkungen
1	Demonstriert einen BUS mit zwei verschiedenen Sendern und einem Empfänger. Der BUS besteht aus einer Leitung und einem Three-State-Puffer-IC 74LS125 mit vier Puffern.
2	Zeigt das Verhalten von vier Invertern 74LS05, deren

offene Kollektorausgänge über einen 1000 Ohm Widerstand mit +5 V verbunden sind.

3 Demonstriert eine ein-Bit-Puffer/Auffang-Register-Schaltung mit dem D-Flipflop 74LS74 sowie einer Pufferstufe.

4 Zeigt eine 8-Bit-Mikrocomputer-Eingangsschaltung mit zwei Auffang-Register 74LS175 mit vier Puffern und zwei Puffer-ICs 74LS365 mit sechs Puffern.

5 Demonstriert Mikroprozessor-I/O mit einem Zähler/Auffang-Register 74LS90 und einer Pufferschaltung 74LS365 für den Eingang sowie einem 4-Bit-Auffang-Register 74LS175 für den Ausgang.

VERSUCH NR. 1

Dieser Versuch verdeutlicht die Arbeitsweise des Puffer-ICs 74LS125 in einer BUS-Schaltung mit zwei Sendern und einem Empfänger. Die Sender sind mit zwei logischen Schaltern auf der Experimentier-Platine des Nanocomputer®-Systems realisiert. Der Empfänger besteht aus einer LED.

1. Schritt

Fertigen Sie die Schaltung nach Bild 4-28 an. 74LS125 enthält vier unabhängige BUS-Puffer. Sie benötigen nur zwei davon, je einen pro Sender.

2. Schritt

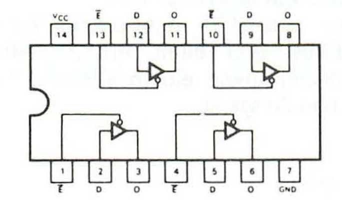
Setzen Sie die Schalter SW1 und SW7 in Stellung logisch 1 und verbinden die Platine mit der Stromversorgung. Betätigen Sie den Schalter SW0, schalten Sie ihn mehrmals von log. 1 nach log. 0 und zurück. Was beobachten Sie?
Die LED LM0 leuchtet nicht auf, die Stellung von SW0 hat keinen Einfluß darauf.

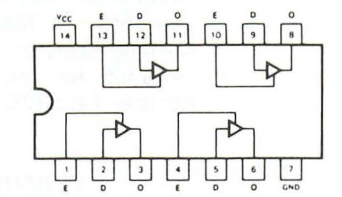
3. Schritt

Schalten sie SW7 aus (OFF) und SW6 mehrmals ein und aus. Was beobachten Sie?
Die LED LM0 leuchtet im Umschaltrhythmus auf. Der Schalter SW7 verbindet also in Stellung log. 0 den Puffereingang mit dem BUS. Der Sender SW6 und der Empfänger LM0 stehen also über den BUS miteinander in Verbindung. Diese Verbindung bleibt solange bestehen, bis man den Puffer wieder blockiert. Dazu muß der Freigabeanschluß logisch 1 werden.
Setzen Sie SW7 in Stellung OFF (log. 0) und SW1 in Stellung ON (log. 1) und betätigen Sie mehrmals den Schalter SW0. Wie reagiert die LED LM0?
Sie reagiert gar nicht. Je nach Stellung von SW6 leuchtet sie oder ist dunkel. Jedenfalls ändert sie nicht den jeweiligen Zustand.

Tabelle 4-10. Eigenschaften des 74LS125/74LS126.

QUAD 3-STATE BUFFERS WITH ACTIVE HIGH ENABLES





GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE MIN	SUPPLY VOLTAGE TYP	SUPPLY VOLTAGE MAX	TEMPERATURE
T54LS125X T54LS126X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS125X T74LS126X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.4	3.4		V	I_{OH} = −1.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or
		74	2.4	3.1		V	I_{OH} = −2.6 mA	V_{IL} per Truth Table
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 12 mA	V_{CC} = MIN, V_{IN} = V_{IH} or
		74		0.35	0.5	V	I_{OL} = 24 mA	V_{IL} per Truth Table
I_{OZH}	Output Off Current HIGH				20	µA	V_{CC} = MAX, V_{OUT} = 2.4 V, V_E = V_{IL}	
I_{OZL}	Output Off Current LOW				−20	µA	V_{CC} = MAX, V_{OUT} = 0.4 V, V_E = V_{IL}	
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 3)		−30		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current, Outputs LOW	LS125			16	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 0 V	
		LS126			20	mA	V_{CC} = MAX, V_{IN} = 0 V, V_E = 4.5 V	
	Power Supply Current, Outputs Off	LS125			20	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 4.5 V	
		LS126			24	mA	V_{CC} = MAX, V_{IN} = 0 V, V_E = 0 V	

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tabelle 4-10. Fortsetzung

TRUTH TABLES

LS125

INPUTS		OUTPUT
$\bar{E}$	D	
L	L	L
L	H	H
H	X	(Z)

LS126

INPUTS		OUTPUT
E	D	
H	L	L
H	H	H
L	X	(Z)

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
(Z) = High Impedance (off)

AC CHARACTERISTICS: $T_A = 25°C$

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS	
		MIN	TYP	MAX			
t_{PLH} t_{PHL}	Propagation Delay, Data to Output			10 16	ns	Fig. 2	$V_{CC} = 5.0$ V $C_L = 45$ pF $R_L = 667\ \Omega$
t_{PZH}	Output Enable Time to HIGH Level			16	ns	Figs. 4, 5	
t_{PZL}	Output Enable Time to LOW Level			30	ns	Figs. 3, 5	
t_{PLZ}	Output Disable Time from LOW Level			15	ns	Figs. 3, 5	$V_{CC} = 5.0$ V $C_L = 5$ pF $R_L = 667\ \Omega$
t_{PHZ}	Output Disable Time from HIGH Level			23	ns	Figs. 4, 5	

Fig. 1

Fig. 2

Fig. 3

Fig. 4

SWITCH POSITIONS

SYMBOL	SW1	SW2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Open
t_{PLZ}	Closed	Closed
t_{PHZ}	Closed	Closed

Fig. 5

Tabelle 4-11. Eigenschaften des 74LS42

T54LS42/T74LS42

ONE-OF-TEN DECODER

DESCRIPTION – The LSTTL/MSI T54LS42/T74LS42 is a Multipurpose Decoder designed to accept four BCD inputs and provide ten mutually exclusive outputs. The LS42 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

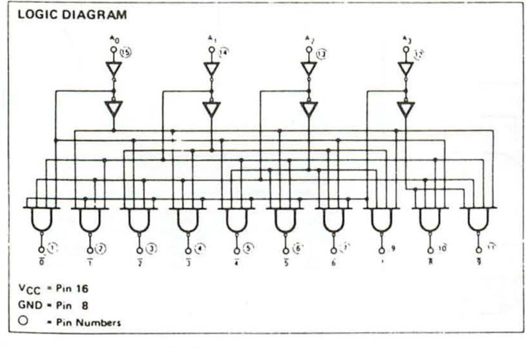
- **MULTI-FUNCTION CAPABILITY**
- **MUTUALLY EXCLUSIVE OUTPUTS**
- **DEMULTIPLEXING CAPABILITY**
- **INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS**
- **FULLY TTL AND CMOS COMPATIBLE**

PIN NAMES		LOADING (Note a) HIGH	LOW
$A_0 - A_3$	Address Inputs	0.5 U.L.	0.25 U.L.
$\overline{0}$ to $\overline{9}$	Outputs, Active LOW (Note b)	10 U.L.	5(2.5) U.L.

NOTES

a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.

b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.



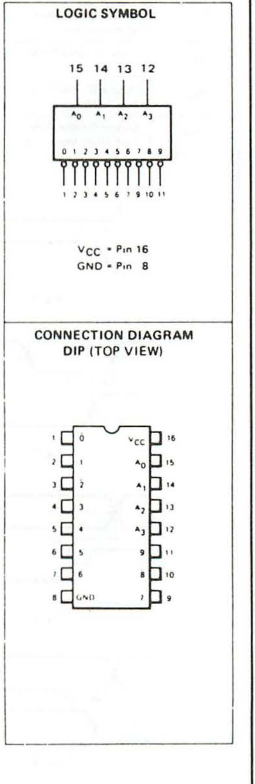


Tabelle 4-11. Fortsetzung

FUNCTIONAL DESCRIPTION – The LS42 decoder accepts four active HIGH BCD inputs and provides ten mutually exclusive active LOW outputs, as shown by logic symbol or diagram. The active LOW outputs facilitate addressing other MSI units with active LOW input enables.

The logic design of the LS42 ensures that all outputs are HIGH when binary codes greater than nine are applied to the inputs.

The most significant input A_3 produces a useful inhibit function when the LS42 is used as a one-of-eight decoder. The A_3 input can also be used as the Data input in an 8-output demultiplexer application.

TRUTH TABLE

A_0	A_1	A_2	A_3	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{5}$	$\bar{6}$	$\bar{7}$	$\bar{8}$	$\bar{9}$
L	L	L	L	L	H	H	H	H	H	H	H	H	H
H	L	L	L	H	L	H	H	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H	H	H
H	H	L	L	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	H	H	H	L	H	H	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H
H	H	H	L	H	H	H	H	H	H	H	L	H	H
L	L	L	H	H	H	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L
L	H	L	H	H	H	H	H	H	H	H	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = HIGH Voltage Level
L = LOW Voltage Level

ABSOLUTE MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	−65°C to +150°C
Temperature (Ambient) Under Bias	−55°C to +125°C
V_{CC} Pin Potential to Ground Pin	−0.5 V to +7.0 V
*Input Voltage (dc)	−0.5 V to +15 V
*Input Current (dc)	−30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	−0.5 V to +10 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or Input Current limit is sufficient to protect the inputs

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			TEMPERATURE
	MIN	TYP	MAX	
T54LS42X	4.5 V	5.0 V	5.5 V	−55°C to +125°C
T74LS42X	4.75 V	5.0 V	5.25 V	0°C to +75°C

X = package type, D for Ceramic Dip, B for Plastic Dip See Packaging Information Section for packages available on this product

Tabelle 4-11. Fortsetzung

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Threshold Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Threshold Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.4				
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 4)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current			7.0	12	mA	V_{CC} = MAX	

NOTES:
1. Conditions for testing, not shown in the Table, are chosen to guarantee operation under "worst case" conditions.
2. The specified LIMITS represent the "worst case" value for the parameter. Since these "worst case" values normally occur at the temperature and supply voltage extremes, additional noise immunity and guard banding can be achieved by decreasing the allowable system operating ranges.
3. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
4. Not more than one output should be shorted at a time.

AC CHARACTERISTICS: T_A = 25°C

SYMBOL	PARAMETER	LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS	
t_{PLH} t_{PHL}	Propagation Delay (2 Levels)		11 18	18 25	ns	Fig. 2	V_{CC} = 5.0 V
t_{PLH} t_{PHL}	Propagation Delay (3 Levels)		12 19	20 27	ns	Fig. 1	C_L = 15 pF

AC WAVEFORMS

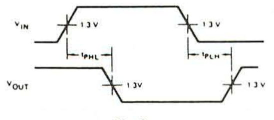


Fig. 1

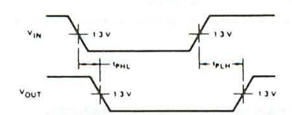


Fig. 2

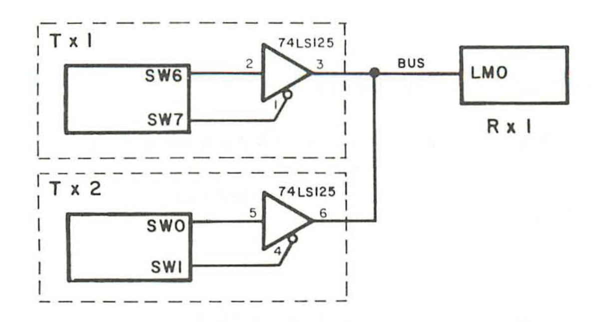


Bild 4-28. Mit einer einzigen BUS-Leitung sind drei Geräte (zwei Sender und ein Empfänger) miteinander verbunden.

4. Schritt

Beantworten Sie theoretisch folgende Frage: Was geschieht, wenn sowohl SW1 und SW7 gleichzeitig logisch 0 sind? NICHT AUSPROBIEREN!

In diesem Fall sind beide Puffer aktiviert, eine *unerwünschte* Situation. Erstens: Beide Sender versuchen ihre Daten über den BUS zum Empfänger zu übertragen. Dadurch werden die Daten nur verstümmelt. Zweitens: Das IC ist nicht für derartige Aufgaben konzipiert, so daß eine Beschädigung nicht auszuschließen ist.

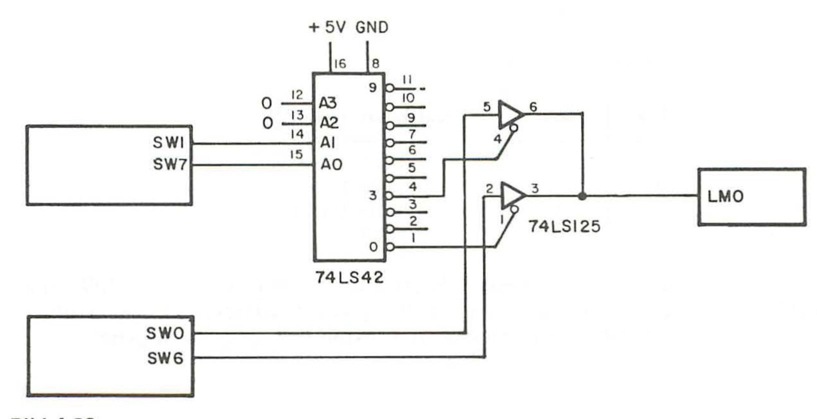


Bild 4-29.

5. Schritt

Füllen Sie die nachstehende Tabelle aus:

SW1	SW7	von LMO angezeigte Daten
0	0	Katastrophe!!
0	1	
1	0	
1	1	

Ihr Beobachtungen müßten die Gültigkeit der folgenden Tabelle bestätigen:

SW1	SW7	von LMO angezeigte Daten
0	0	Katastrophe!!
0	1	SW0
1	0	SW6
1	1	keine Daten

6. Schritt

Ändern Sie Ihre Schaltung gemäß Bild 4-29. Das IC 74LS42 ist ein 1-zu-10 Dekoder (BCD-zu-Dezimal). Die Pinbelegung mit der entsprechenden Wahrheitstabelle entnehmen Sie der Tabelle 4-11.
Welche Aufgabe erfüllt das Dekoder-IC 74LS42.
Die Aufgabe besteht darin, nur immer einen Puffer freizugeben. Versuchen Sie nachfolgende Tabelle theoretisch zu ergänzen und vergleichen Sie das Ergebnis mit der komplett ausgefüllten Tabelle:

SW1	SW7	von LMO angezeigte Daten
0	0	
0	1	
1	0	
1	1	

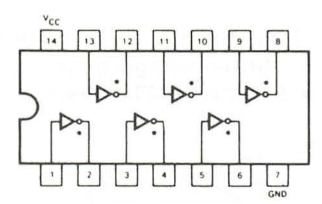
So sieht die richtige Tabelle aus:

SW1	SW7	Daten auf dem LMO
0	0	SW6
0	1	keine Daten
1	0	keine Daten
1	1	SW0

Ein Hauptvorteil: Es gibt keine gleichzeitige Kombination von SW1 und SW7, welche den Bauteilen in der Schaltung einen Schaden zufügen könnte. Daher ist dieser Schaltung gegenüber der ersten den Vorzug zu geben.

Tabelle 4-12. Eigenschaften des 74LS05

HEX INVERTER



*OPEN COLLECTOR OUTPUTS

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS05X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS05X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS (Note 1)
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
I_{OH}	Output HIGH Current				100	µA	V_{CC} = MIN, V_{OH} = 5.5 V, V_{IN} = V_{IL}
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 5.5 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{CCH}	Supply Current HIGH			1.2	2.4	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			3.6	6.6	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C (See Page 4-50 for Waveforms)

SYMBOL	PARAMETER	LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS
t_{PLH}	Turn Off Delay, Input to Output		14	22	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output		10	18	ns	C_L = 15 pF, R_L = 2.0 kΩ

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.

VERSUCH NR. 2

Dieser Versuch demonstriert das Verhalten der vier Inverter 74LS05, deren offene Kollektorausgänge über einen 1000 Ohm Widerstand mit +5V verbunden sind.

1. Schritt

Bauen Sie die in Bild 4-30 gezeigte Schaltung auf. Legen Sie an die Experimentier-Platine die Strombesorgung an und stellen die Schalter SW0 . . . SW3 auf log. 0. Leuchtet LM0 auf oder nicht?

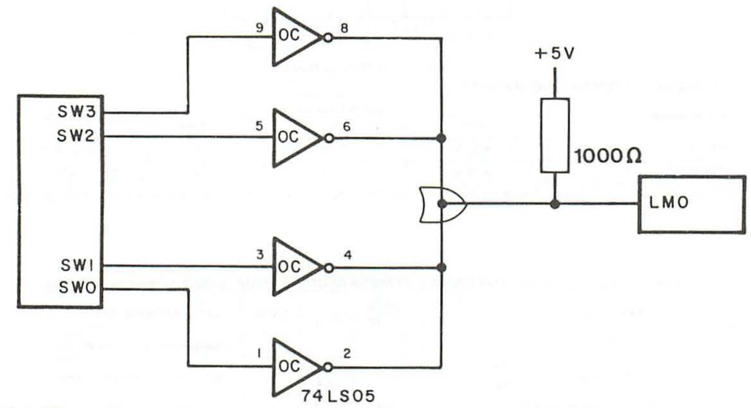


Bild 4-30.

Die LED LM0 leuchtet auf. (Der gemeinsame Inverterausgang ist logisch 1. Dieser Signalpegel wird im LED-Monitor invertiert.)

2. Schritt

Setzen Sie SW0 auf logisch 1, während sich SW1, SW2 und SW3 im Zustand logisch 0 befinden. Wie reagiert die LED LM0?

Die LED LM0 erlischt.

3. Schritt

Füllen Sie die folgende Wahrheitstabelle aus, wobei die aufleuchtende LED LM0 = logisch 1 und die dunkele = logisch 0 zu setzen ist.

SW0	SW1	SW2	SW3	LM0
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Zu welcher Gatterart mit vier Eingängen gehört die obige Wahrheitstabelle?
Die Antwort lautet: zu einem NOR-Gatter mit vier Eingängen.

VERSUCH NR. 3

Dieser Versuch beschäftigt sich mit Puffer-Auffang-Register-Techniken. Sie lernen ein 1-Bit-Puffer/Auffang-Register kennen, das mit dem D-Flipflop 74LS74 und dem Puffertreiber 74LS125 aufgebaut ist.

1. Schritt

Bauen Sie die in Bild 4-31 gezeigte Schaltung auf. Verbinden Sie die Experimentier-Platine mit der Stromversorgung und stellen den Schalter SW0 auf ON (logisch 1). Stellen Sie fest, welcher Logikpegel den Puffer 74LS125 blockiert und welche Impulsflanke (Übergang von "0" nach "1" bzw. von "1" nach "0") das Flipflop 74LS74 taktet.
Bei dem Logikpegel von logisch 1 ist der Puffer blockiert; während eine positive Impulsflanke (Übergang von "0" nach "1") das Flipflop taktet.

2. Schritt

Prüfen Sie folgende Wahrheitstabelle mit der aus Bild 4-31 aufgebauten Schaltung nach:

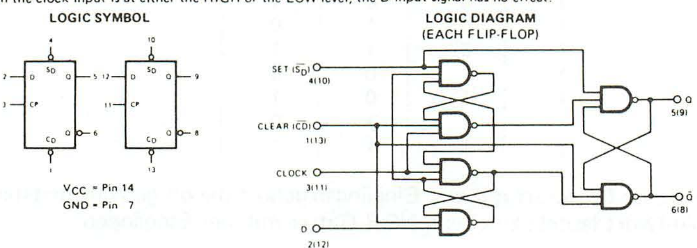
SW0	SW1	P0	Ausgang 74LS74 (LM0)	Ausgang 74LS125 (LM1)
0	X	0	P	P = vorher aufgefangene Daten
0	0	0-1-0 (⎍)	0	0
0	1	0-1-0 (⎍)	1	1
1	X	0	P	hochohmiger Zustand
1	0	0-1-0 (⎍)	0	hochohmiger Zustand
1	1	0-1-0 (⎍)	1	hochohmiger Zustand

Tabelle 4-13. Eigenschaften des Dual-Flipflops 74LS74

DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOP

DESCRIPTION – The T54LS74/T74LS74 dual edge-triggered flip-flop utilizes Schottky TTL circuitry to produce high speed D-type flip-flops. Each flip-flop has individual clear and set inputs, and also complementary Q and $\bar{Q}$ outputs.

Information at input D is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the HIGH or the LOW level, the D input signal has no effect.



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS74X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS74X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type, D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS (Note 1)	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.4				
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or
		74		0.35	0.5	V	I_{OL} = 8.0 mA	V_{IL} per Truth Table
I_{IH}	Input HIGH Current Data Clock, Set Clear				20 40 60	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
	Data Clock, Set Clear				0.1 0.2 0.3	mA	V_{CC} = MAX, V_{IN} = 5.5 V	
I_{IL}	Input LOW Current Data Clock, Set Clear				−0.4 −0.8 −1.2	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 3)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current			4.0	8.0	mA	V_{CC} = MAX, V_{CP} = 0 V	

Tabelle 4-13. Fortsetzung

MODE SELECT – TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	$\overline{S}_D$	$\overline{C}_D$	D	Q	$\overline{Q}$
Set	L	H	X	H	L
Reset (Clear)	H	L	X	L	H
*Undetermined	L	L	X	H	H
Load "1" (Set)	H	H	h	H	L
Load "0" (Reset)	H	H	l	L	H

*Both outputs will be HIGH while both $\overline{S}_D$ and $\overline{C}_D$ are LOW, but the output states are unpredictable if $\overline{S}_D$ and $\overline{C}_D$ go HIGH simultaneously.

H,h = HIGH Voltage Level
L,l = LOW Voltage Level
X = Don't Care
l, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition.

AC CHARACTERISTICS: $T_A = 25°C$ (See Page 4-51 for Waveforms)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS	
			MIN	TYP	MAX			
f_{MAX}	Maximum Clock Frequency		30	45		MHz	Fig. 1	$V_{CC} = 5.0$ V, $C_L = 15$ pF
t_{PLH} t_{PHL}	Propagation Delay, Clock to Output			15 22	20 30	ns	Fig. 1	
t_{PLH}	Propagation Delay, Set or Clear to Output			10	15	ns	Fig. 2	
t_{PHL}		CP = L		18	24			
t_{PHL}		CP = H		26	35			

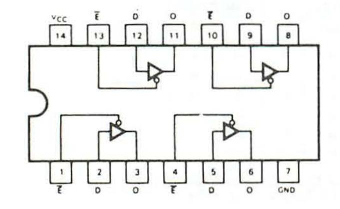
AC SET-UP REQUIREMENTS: $T_A = 25°C$ (See Page 4-51 for Waveforms)

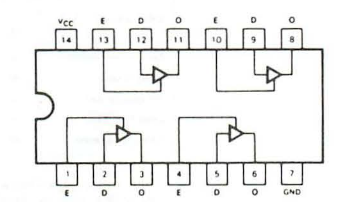
SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS	
		MIN	TYP	MAX			
$t_w CP(H)$	Clock Pulse Width (HIGH)	18	12		ns	Fig. 1	$V_{CC} = 5.0$ V
t_w	Set or Clear Pulse Width	15	10		ns	Fig. 2	
$t_s(H)$	Set-up Time HIGH, Data to Clock	10	6		ns	Fig. 1	
$t_h(H)$	Hold Time HIGH, Data to Clock	0	−14		ns		
$t_s(L)$	Set-up Time LOW, Data to Clock	20	14		ns		
$t_h(L)$	Hold Time LOW, Data to Clock	0	−6		ns		

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at $V_{CC} = 5.0$ V, $T_A = 25°C$.
3. Not more than one output should be shorted at a time.
4. SET-UP TIME (t_s) is defined as the minimum time required for the correct logic level to be present at the logic input prior to the clock transition from LOW to HIGH in order to be recognized and transferred to the outputs.
5. HOLD TIME (t_h) is defined as the minimum time following the clock transition from LOW to HIGH that the logic level must be maintained at the input in order to ensure continued recognition. A negative HOLD TIME indicates that the correct logic level may be released prior to the clock transition from LOW to HIGH and still be recognized.

Tabelle 4-14. Eigenschaften des Vierfach-Puffers 74LS125

QUAD 3-STATE BUFFERS WITH ACTIVE HIGH ENABLES





GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS125X T54LS126X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS125X T74LS126X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.4	3.4		V	I_{OH} = −1.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.4	3.1		V	I_{OH} = −2.6 mA	
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 12 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 24 mA	
I_{OZH}	Output Off Current HIGH				20	µA	V_{CC} = MAX, V_{OUT} = 2.4 V, V_E = V_{IL}	
I_{OZL}	Output Off Current LOW				−20	µA	V_{CC} = MAX, V_{OUT} = 0.4 V, V_E = V_{IL}	
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 3)		−30		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current, Outputs LOW	LS125			16	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 0 V	
		LS126			20	mA	V_{CC} = MAX, V_{IN} = 0 V, V_E = 4.5 V	
	Power Supply Current, Outputs Off	LS125			20	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 4.5 V	
		LS126			24	mA	V_{CC} = MAX, V_{IN} = 0 V, V_E = 0 V	

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25° C.
3. Not more than one output should be shorted at a time.

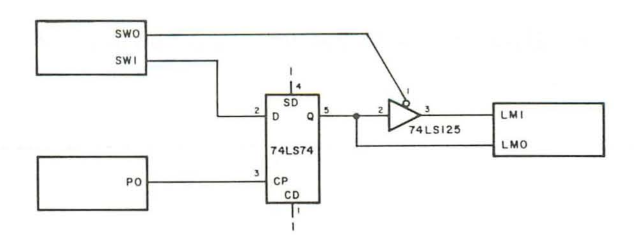


Bild 4-31.

Der Logikpegel am D-Eingang wird nur dann zum Q-Ausgang übertragen, wenn der CP-Eingang einen Taktimpuls erhält. Der Übertrag vom Ein- zum Ausgang wird von der positiven Impulsflanke ausgelöst. Wenn kein Übergang von "0" nach "1" am Takteingang erfolgt, sind die Logikpegel am Aus- und Eingang nicht unbedingt identisch.
Wenn der Q-Ausgang einmal gesetzt ist, bleibt er unabhängig vom D-Eingang konstant. Das verdeutlicht die Unabhängigkeit von D und Q. Darüber hinaus sind auch der von dem LM0 überwachte Ausgang 74LS74 und der von LM1 kontrollierte Ausgang 74LS125 unabhängig. LM1 zeigt den Zustand am Q-Ausgang des Flipflops nur dann an, wenn der Puffer freigegeben ist. So können Sie durch sinnvolle Manipulation von SW0 GENAU kontrollieren, wie lange der Flipflop-Ausgang an LM1 oder einen BUS angeschlossen bleiben darf.

3. Schritt

Mit der in Bild 4-31 gezeigten Schaltung sollen Sie ein Bit für die CPU zum Abruf bereitstellen, wissen aber nicht, wann die CPU das Bit abliest. Was müssen Sie tun?
Folgende Problemlösung ist möglich:
Schritt 1: Stellen Sie mit SW1 das gewünschte Bit ein ("1" oder "0").
Schritt 2: Lösen Sie mit P0 einen Taktimpuls aus, so daß das Bit vom D-Eingang auf den Q-Ausgang übertragen wird.
Selbst wenn Sie jetzt die Stellung von SW1 ändern, bleibt der Ausgangszustand erhalten, da kein weiterer Taktimpuls folgt.
Schritt 3: Das Bit steht jetzt zum Abruf durch die CPU bereit. Sobald das CPU-Signal den Puffer freigibt, wird das Bit zum Pufferausgang übertragen. Simulieren Sie das CPU-Signal durch Betätigen des Schalters SW0 von OFF nach ON und wieder nach OFF.

VERSUCH NR. 4

Der folgende Versuch zeigt die Funktion einer 8-Bit-Mikrocomputer-Eingabeschaltung mit Hilfe von zwei Flipflop-ICs 74LS175 mit vier Puffern für das Auffang-Register und zwei Puffer-ICs 74LS365 mit sechs Puffern.

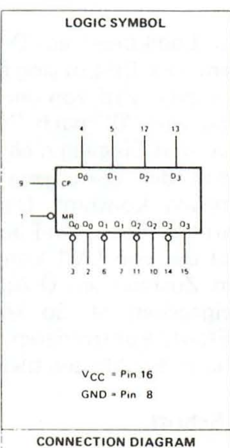
Tabelle 4-15. Eigenschaften des Flipflops 74LS175 mit vier Puffern

DESCRIPTION – The LSTTL/MSI T54LS175/T74LS175 is a high speed Quad D Flip-Flop. The device is useful for general flip-flop requirements where clock and clear inputs are common. The information on the D inputs is stored during the LOW to HIGH clock transition. Both true and complemented outputs of each flip-flop are provided. A Master Reset input resets all flip-flops, independent of the Clock or D inputs, when LOW.

The LS175 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- EDGE-TRIGGERED D-TYPE INPUTS
- BUFFERED-POSITIVE EDGE-TRIGGERED CLOCK
- CLOCK TO OUTPUT DELAYS OF 14 ns
- ASYNCHRONOUS COMMON RESET
- TRUE AND COMPLEMENT OUTPUT
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

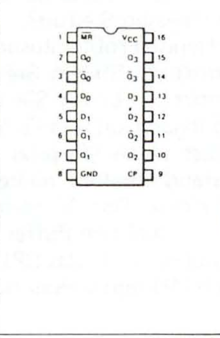
LOGIC SYMBOL



PIN NAMES		LOADING (Note a) HIGH	LOW
$D_0 - D_3$	Data Inputs	0.5 U.L.	0.25 U.L.
CP	Clock (Active HIGH Going Edge) Input	0.5 U.L.	0.25 U.L.
$\overline{MR}$	Master Reset (Active LOW) Input	0.5 U.L.	0.25 U.L.
$Q_0 - Q_3$	True Outputs (Note b)	10 U.L.	5(2.5) U.L.
$\overline{Q}_0 - \overline{Q}_3$	Complemented Outputs (Note b)	10 U.L.	5(2.5) U.L.

NOTES
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

CONNECTION DIAGRAM
DIP (TOP VIEW)



LOGIC DIAGRAM

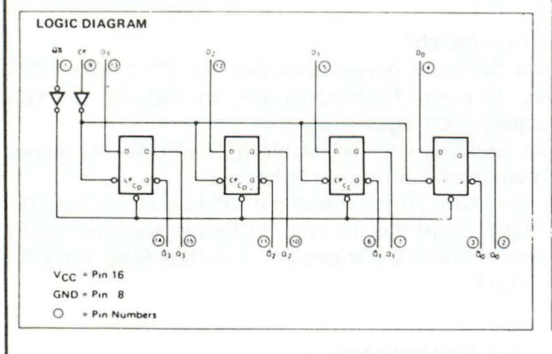


Tabelle 4-15. Fortsetzung

FUNCTIONAL DESCRIPTION – The LS175 consists of four edge-triggered D flip-flops with individual D inputs and Q and $\overline{Q}$ outputs. The Clock and Master Reset are common. The four flip-flops will store the state of their individual D inputs on the LOW to HIGH Clock (CP) transition, causing individual Q and $\overline{Q}$ outputs to follow. A LOW input on the Master Reset ($\overline{MR}$) will force all Q outputs LOW and $\overline{Q}$ outputs HIGH independent of Clock or Data inputs.

The LS175 is useful for general logic applications where a common Master Reset and Clock are acceptable.

TRUTH TABLE

Inputs (t = n, $\overline{MR}$ = H)	Outputs (t = n+1) Note 1	
D	Q	$\overline{Q}$
L	L	H
H	H	L

Note 1: t = n + 1 indicates conditions after next clock.

ABSOLUTE MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	−65°C to +150°C
Temperature (Ambient) Under Bias	−55°C to +125°C
V_{CC} Pin Potential to Ground Pin	−0.5 V to +7.0 V
*Input Voltage (dc)	−0.5 V to +15 V
*Input Current (dc)	−30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	−0.5 V to +10 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or Input Current limit is sufficient to protect the inputs.

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			TEMPERATURE
	MIN	TYP	MAX	
T54LS175X	4.5 V	5.0 V	5.5 V	−55°C to +125°C
T74LS175X	4.75 V	5.0 V	5.25 V	0°C to +75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

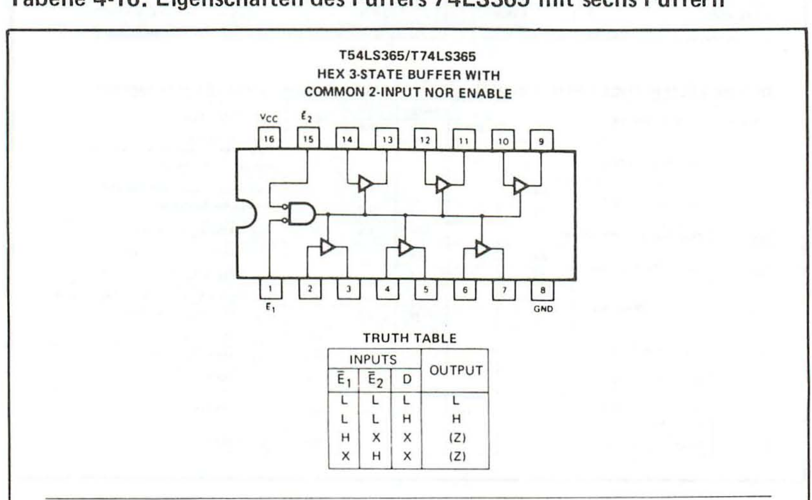
SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Threshold Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Threshold Voltage for All Inputs
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA; V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{SC}	Output Short Circuit Current (Note 4)		−15		100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CC}	Power Supply Current			11	18	mA	V_{CC} = MAX

Programm UCINP

```
Objekt-Code                Quell-Code              Bemerkung

                           NAME UCINP
D311          UCINP:       OUT (11H),A             ;Daten von Schaltern auf-
                                                   ;fangen
CD9A01                     CALL WAIT               ;eine Weile verzögern
0E12                       LD C,12H                ;Eingangsgerätecode in C-
                                                   ;Register setzen
ED40                       IN B,(C)                ;durch Freigabe des
                                                   ;Puffers Daten vom Auf-
                                                   ;fang-Register ins B-
                                                   ;Register eingeben
FF                         RST 38H                 ;Steuerung zum Opera-
                                                   ;tionssystem des Nano-
                                                   ;computers® zurück-
                                                   ;geben
210500        WAIT:        LD HL, 0005H            ;Verzögerungsschleife
11FFFF        LOOP5:       LD DE, 0FFFFH
1B            LOOP6:       DEC DE
7A                         LD A, D
B3                         OR E
20FB                       JR NZ, LOOP6
2B                         DEC HL
7D                         LD A,L
B4                         OR H
20F3                       JR NZ,LOOP5
C9                         RET
```

Tabelle 4-16. Eigenschaften des Puffers 74LS365 mit sechs Puffern



T54LS365/T74LS365
HEX 3-STATE BUFFER WITH
COMMON 2-INPUT NOR ENABLE

TRUTH TABLE

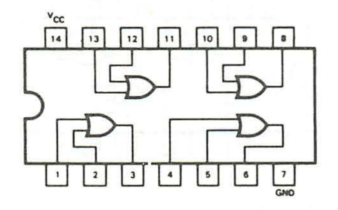
INPUTS			OUTPUT
$\bar{E}_1$	$\bar{E}_2$	D	
L	L	L	L
L	L	H	H
H	X	X	(Z)
X	H	X	(Z)

DESCRIPTION – The LS365/366/367/368 are high speed hex buffers with 3-state outputs. They are organized as single 6-bit or 2-bit/4-bit, with inverting or non-inverting data (D) paths. The outputs are designed to drive 15 TTL Unit Loads or 60 Low Power Schottky loads when the Enable ($\bar{E}$) is LOW.

When the Output Enable Input ($\bar{E}$) is HIGH, the outputs are forced to a high impedance "off" state. If the outputs of the 3-state devices are tied together, all but one device must be in the high impedance state to avoid high currents that would exceed the maximum ratings. Designers should ensure that Output Enable signals to 3-state devices whose outputs are tied together are designed so there is no overlap.

Tabelle 4-17. Eigenschaften des Gatters 74LS32

QUAD 2-INPUT OR GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS32X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS32X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA, V_{IN} = V_{IH}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = V_{IL}
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = V_{IL}
I_{IH}	Input HIGH Current			1.0	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			3.1	6.2	mA	V_{CC} = MAX, Inputs Open
I_{CCL}	Supply Current LOW			4.9	9.8	mA	V_{CC} = MAX, V_{IN} = 0 V

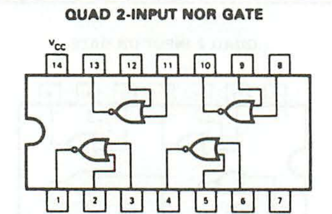
AC CHARACTERISTICS: T_A = 25°C (See Page 4-50 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	7.0	11	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	7.0	11	ns	C_L = 15 pF

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tabelle 4-18. Eigenschaften des Gatters 74LS02

QUAD 2-INPUT NOR GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS02X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS02X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

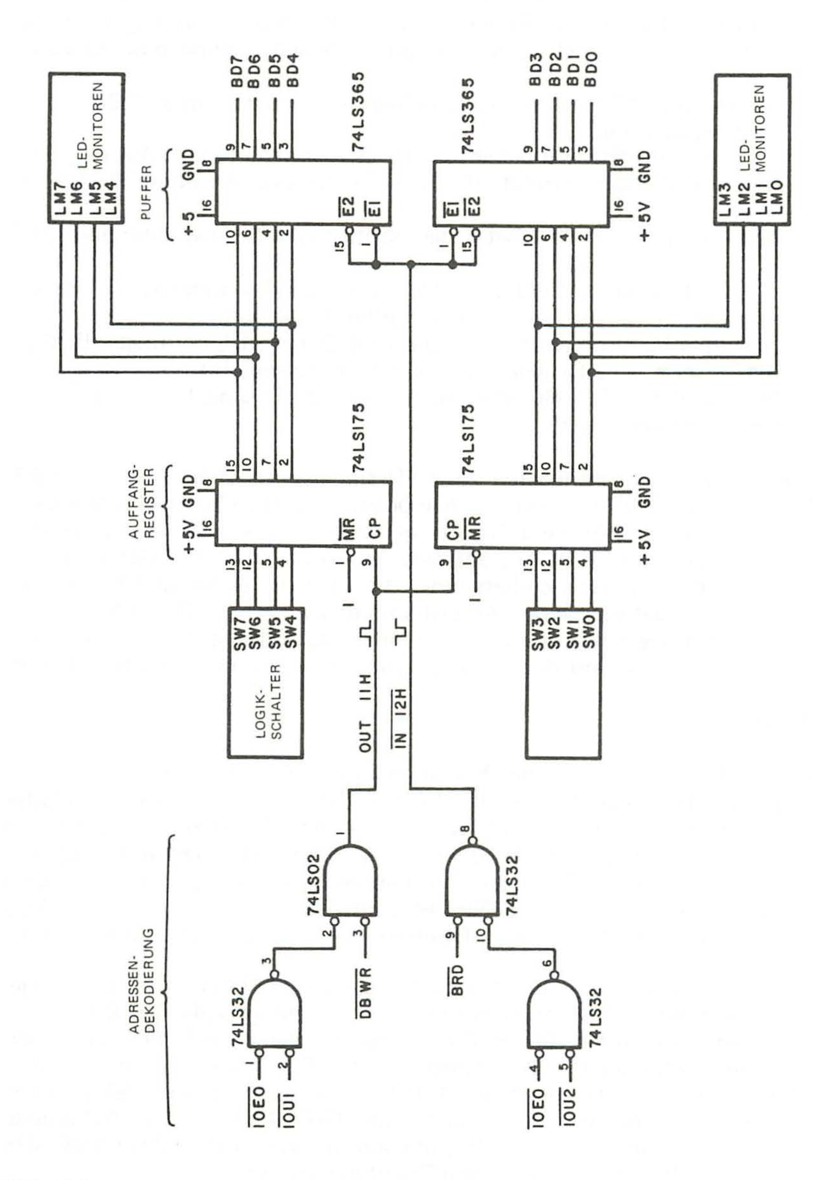
DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS (Note 1)
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 μA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			1.6	3.2	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			2.4	5.4	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C (See Page 4-50 for Waveforms)

SYMBOL	PARAMETER	LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS
t_{PLH}	Turn Off Delay, Input to Output	3.0	5.0	10	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	5.0	10	ns	C_L = 15 pF

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.



LM7
LM6
LM5
LM4
LED-MONITOREN
PUFFER
+5
GND
BD7
BD6
BD5
BD4
74LS365
E2
E1
74LS365
BD3
BD2
BD1
BD0
E1
E2
+5V
GND
LM3
LM2
LM1
LM0
LED-MONITOREN
AUFFANG-REGISTER
+5V
GND
74LS175
MR
CP
74LS175
CP
MR
+5V
GND
SW7
SW6
SW5
SW4
LOGIK-SCHALTER
OUT 11H
IN 12H
SW3
SW2
SW1
SW0
ADRESSEN-DEKODIERUNG
74LS02
74LS32
74LS32
DB WR
BRD
74LS32
IOE0
IOU1
IOE0
IOU2

Bild 4-32.

1. Schritt

Bauen Sie die in Bild 4-32 dargestellte Schaltung. Im Prinzip ist diese Schaltung lediglich eine Erweiterung der Puffer/Auffang-Register-Schaltung aus dem 3. Versuch. Es sind lediglich einige Änderungen notwendig:

a) Es ist ein 8-Bit-Puffer/Auffang-Register anstatt eines 1-Bit-Puffer/Auffang-Registers.
b) Der Puffer-Freigabe/Blockierschalter SW0 ist durch den Ausgang einer Dekoderschaltung ersetzt, die den Geräte-Auswahlimpuls $\overline{\text{IN 12H}}$ erzeugt.
c) Den Impulsgeber P0 ersetzt der positive Geräte-Auswahlimpuls OUT 11H.
d) Die acht Schalter SW0 . . . SW7 ersetzen das Eingangs-D-Flipflop. Ansonsten wäre pro Schalter ein Flipflop notwendig.
e) Der LED-Monitor wird durch die acht Daten-BUS-Leitungen BD0 bis BD7 ersetzt, eine für jedes von der CPU zu lesende Bit.
f) Anstelle der LED LM0 überprüfen alle acht LEDs die 8 Q-Ausgänge am 8-Bit-Auffang-Register.

Die Quelle für Eingabedaten zur CPU sind die Schalter SW0 . . . SW7. Die Daten werden zunächst mit dem positiven Geräte-Auswahlimpuls OUT 11H in das 8-Bit-Auffang-Register getastet. Dort kann man sie für eine beliebige Zeit speichern, bis entweder ein neuer OUT 11H-Befehl andere Daten in die Flipflops einliest, oder die CPU die Daten abruft. Sie gibt über einen negativen Geräte-Auswahlimpuls $\overline{\text{IN 12H}}$ die Daten frei. Durch das Ablesen beeinflußt die CPU den Inhalt des Auffang-Registers natürlich nicht. Die Daten sind dort solange gespeichert, bis man sie überschreibt.

2. Schritt

Laden Sie das vorstehende Programm in den Speicher des Nanocomputers®, beginnend an der Speicherstelle UCINP. Laden Sie 00 ins B-Register und stellen Sie die logischen Schalter auf hex 75. Definieren Sie einen Ausführungs-Haltepunkt, der unmittelbar hinter der Verzögerungsschleife beim Befehl LD C, 12H liegen soll. Führen Sie das Programm aus, indem Sie an der Speicherstelle UCINP beginnen. Welchen Wert zeigen die LEDs an und was ist der Inhalt des B-Registers, nachdem der Haltepunkt erreicht ist?
Die LEDs zeigen den vom Geräte-Auswahlimpuls OUT 11H aufgefangenen Inhalt des 8-Bit-Auffang-Registers an; er beträgt 75H. Der B-Registerinhalt ist noch immer 00, weil die CPU die Daten noch nicht abgelesen hat. Der Akkumulatorinhalt spielt bei der Befehlsausführung von OUT (11H),A an der Speicherstelle UCINP keine Rolle. Dieser Befehl nutzt nur den erzeugten Geräte-Auswahlimpuls OUT 11H aus. Der Akkumulatorinhalt wird auf den Daten-BUS und auf die obere Hälfte des Adreß-BUS gelegt, jedoch von keiner externen Schaltung benutzt.

3. Schritt

Führen Sie das Programm weiter aus. Überprüfen Sie nach der Beendigung des Programms den Inhalt des B-Registers, es muß das Byte 75H enthalten. Die LEDs zeigen nach wie vor den Inhalt des Auffang-Registers an; sie haben sich also nicht verändert. Die CPU liest das Auffang-Register *zerstörungsfrei* ab, d.h. der Inhalt des Auffang-Registers bleibt trotz der Leseoperation erhalten.

4. Schritt

Ändern Sie den Befehl RST 38H in JP UCINP um. Dadurch tritt die CPU in eine Schleife ein, welche die Eingangssignale des Auffang-Registers mit einbezieht. Wird während der Ausführung der Verzögerungsschleife durch Ändern der Schalterstellungen von SW0 . . . SW7 eine Änderung der Eingangssignale vorgenommen, zeigen die LEDs dies erst an, wenn die CPU den Befehl OUT (11H),A ausführt. Unterbrechen Sie die Schleife durch Drücken der BREAK-Taste (nicht die RESET-Taste benutzen, da sich sonst die Registerinhalte verändern). Je nachdem zu welcher Zeit man die Verzögerungsschleife unterbricht, werden die unterschiedlichen Verhältnisse zwischen den Schaltern, den LEDs und dem B-Registerinhalt deutlich.

LASSEN SIE DIE SUBROUTINE WAIT GELADEN, SIE WIRD IM NÄCHSTEN VERSUCH BENÖTIGT.

VERSUCH NR. 5

Dieser Versuch zeigt, wie eine einfache Mikrocomputer-I/O-Schaltung eine Taktgeberimpulsschaltung kontrolliert. Die Taktimpulse werden von einem Dekadenzähler 74LS90 gezählt und aufgefangen. Die CPU liest die gezählten Impulse in regelmäßigen Abständen, indem sie die Puffer in der Puffer/Auffang-Register-Schaltung freigibt. Das Zählergebnis wird mit vier LEDs im Monitor angezeigt.

Programm UCINM

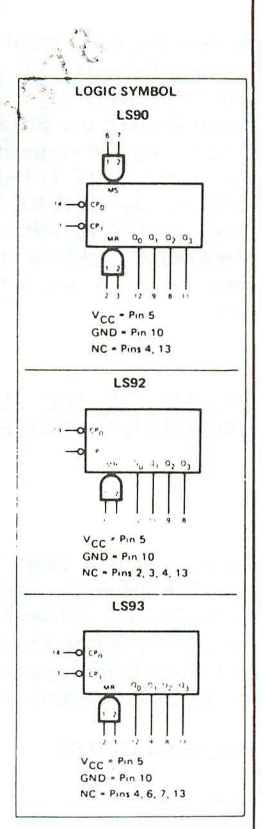
Objekt-Code		Quell-Code	Bemerkung
		NAME UCINM	
0E13	UCINM:	LD C, 13H	;13 als Geräte-Code C
ED40	PCNTR:	IN B,(C)	;Impulszahl in Register ;B eingeben
ED41		OUT (C),B	;Impulszahl an LED- ;Monitoren ausgeben
CD9AD1		CALL WAIT	;vor nächster Zahlable- ;sung verzögern
18F7		JR PCNTR	;Lese/Schreib/Wartezy- ;klus wiederholen

Tabelle 4-19. Eigenschaften des 74LS90

4-BIT BINARY COUNTER

DESCRIPTION – The T54LS90/T74LS90, T54LS92/T74LS92 and T54LS93/T74LS93 are high-speed 4-bit ripple type counters partitioned into two sections. Each counter has a divide-by-two section and either a divide-by-five (LS90), divide-by-six (LS92) or divide-by-eight (LS93) section which are triggered by a HIGH-to-LOW transition on the clock inputs. Each section can be used separately or tied together (Q to $\overline{CP}$) to form BCD, bi-quinary, modulo-12, or modulo-16 counters. All of the counters have a 2-input gated Master Reset (Clear), and the LS90 also has a 2-input gated Master Set (Preset 9).

- **LOW POWER CONSUMPTION . . . TYPICALLY 45 mW**
- **HIGH COUNT RATES . . . TYPICALLY 50 MHz**
- **CHOICE OF COUNTING MODES . . . BCD, BI-QUINARY, DIVIDE-BY-TWELVE, BINARY**
- **INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS**
- **FULLY TTL AND CMOS COMPATIBLE**



PIN NAMES

		LOADING (Note a) HIGH	LOW
$\overline{CP}_0$	Clock (Active LOW going edge) Input to ÷2 Section	3.0 U.L.	1.5 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷5 Section (LS90), ÷6 Section (LS92)	2.0 U.L.	2.0 U.L.
$\overline{CP}_1$	Clock (Active LOW going edge) Input to ÷8 Section (LS93)	1.0 U.L.	1.0 U.L.
MR_1, MR_2	Master Reset (Clear) Inputs	0.5 U.L.	0.25 U.L.
MS_1, MS_2	Master Set (Preset-9, LS90) Inputs	0.5 U.L.	0.25 U.L.
Q_0	Output from ÷2 Section (Notes b & c)	10 U.L.	5(2.5) U.L.
Q_1, Q_2, Q_3	Outputs from ÷5 (LS90), ÷6 (LS92), ÷8 (LS93) Sections (Note b)	10 U.L.	5(2.5) U.L.

NOTES
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.
c. The Q_0 Outputs are guaranteed to drive the full fan-out plus the $\overline{CP}_1$ input of the device.

Tabelle 4-19. Fortsetzung

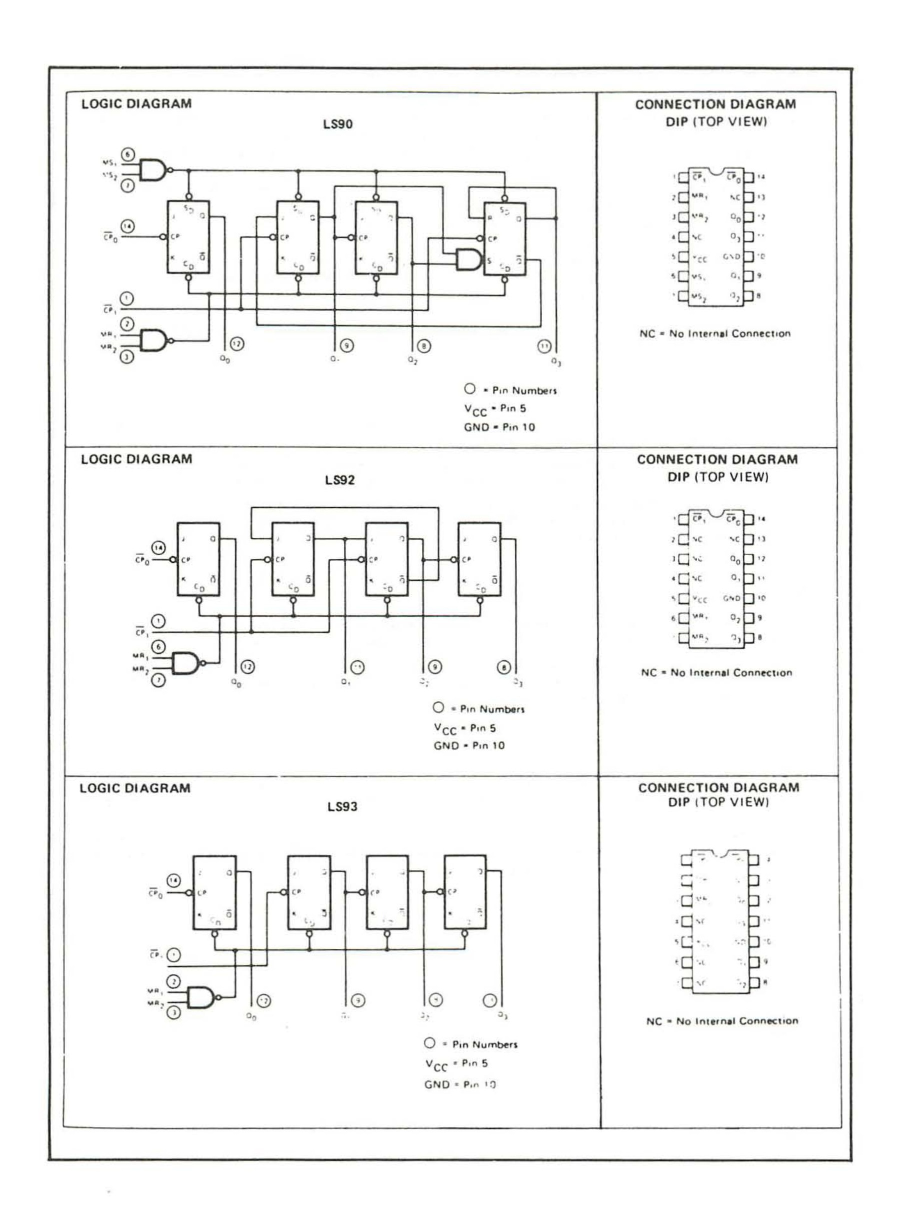


Tabelle 4-19. Fortsetzung

FUNCTIONAL DESCRIPTION – The LS90, LS92, and LS93 are 4-bit ripple type Decade, Divide-By-Twelve, and Binary Counters respectively. Each device consists of four master/slave flip-flops which are internally connected to provide a divide-by-two section and a divide-by-five (LS90), divide-by-six (LS92), or divide-by-eight (LS93) section. Each section has a separate clock input which initiates state changes of the counter on the HIGH-to-LOW clock transition. State changes of the Q outputs do not occur simultaneously because of internal ripple delays. Therefore, decoded output signals are subject to decoding spikes and should not be used for clocks or strobes. The Q_0 output of each device is designed and specified to drive the rated fan-out plus the $\overline{CP}_1$ input of the device.

A gated AND asynchronous Master Reset ($MR_1 \cdot MR_2$) is provided on all counters which overrides and clocks and resets (clears) all the flip-flops. A gated AND asynchronous Master Set ($MS_1 \cdot MS_2$) is provided on the LS90 which overrides the clocks and the MR inputs and sets the outputs to nine (HLLH).

Since the output from the divide-by-two section is not internally connected to the succeeding stages, the devices may be operated in various counting modes.

LS90

A. BCD Decade (8421) Counter – The $\overline{CP}_1$ input must be externally connected to the Q_0 output. The $\overline{CP}_0$ input receives the incoming count and a BCD count sequence is produced.

B. Symmetrical Bi-quinary Divide-By-Ten Counter – The Q_3 output must be externally connected to the $\overline{CP}_0$ input. The input count is then applied to the $\overline{CP}_1$ input and a divide-by-ten square wave is obtained at output Q_0.

C. Divide-By-Two and Divide-By-Five Counter – No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function ($\overline{CP}_0$ as the input and Q_0 as the output). The $\overline{CP}_1$ input is used to obtain binary divide-by-five operation at the Q_3 output.

LS92

A. Modulo 12, Divide-By-Twelve Counter – The $\overline{CP}_1$ input must be externally connected to the Q_0 output. The $\overline{CP}_0$ input receives the incoming count and Q_3 produces a symmetrical divide-by-twelve square wave output.

B. Divide-By-Two and Divide-By-Six Counter – No external interconnections are required. The first flip-flop is used as a binary element for the divide-by-two function. The $\overline{CP}_1$ input is used to obtain divide-by-three operation at the Q_1 and Q_2 outputs and divide-by-six operation at the Q_3 output.

LS93

A. 4-Bit Ripple Counter – The output Q_0 must be externally connected to input $\overline{CP}_1$. The input count pulses are applied to input $\overline{CP}_0$. Simultaneous divisions of 2, 4, 8, and 16 are performed at the Q_0, Q_1, Q_2, and Q_3 outputs as shown in the truth table.

B. 3-Bit Ripple Counter – The input count pulses are applied to input $\overline{CP}_1$. Simultaneous frequency divisions of 2, 4, and 8 are available at the Q_1, Q_2, and Q_3 outputs. Independent use of the first flip-flop is available if the reset function coincides with reset of the 3-bit ripple-through counter.

Tabelle 4-19. Fortsetzung

LS90
MODE SELECTION

RESET/SET INPUTS				OUTPUTS			
MR_1	MR_2	MS_1	MS_2	Q_0	Q_1	Q_2	Q_3
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
L	X	L	X	Count			
X	L	X	L	Count			
L	X	X	L	Count			
X	L	L	X	Count			

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

LS92 AND LS93
MODE SELECTION

RESET INPUTS		OUTPUTS			
MR_1	MR_2	Q_0	Q_1	Q_2	Q_3
H	H	L	L	L	L
L	H	Count			
H	L	Count			
L	L	Count			

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

LS90
BCD COUNT SEQUENCE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H

NOTE: Output Q_0 is connected to Input CP_1 for BCD count.

LS92
TRUTH TABLE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	L	L	H
7	H	L	L	H
8	L	H	L	H
9	H	H	L	H
10	L	L	H	H
11	H	L	H	H

Note: Output Q_0 connected to input CP_1

LS93
TRUTH TABLE

COUNT	OUTPUT			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H
10	L	H	L	H
11	H	H	L	H
12	L	L	H	H
13	H	L	H	H
14	L	H	H	H
15	H	H	H	H

Note: Output Q_0 connected to input CP_1.

ABSOLUTE MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	-65°C to +150°C
Temperature (Ambient) Under Bias	-55°C to +125°C
V_{CC} Pin Potential to Ground Pin	-0.5 V to +7.0 V
*Input Voltage (dc)	-0.5 V to +15 V
*Input Current (dc)	-30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	-0.5 V to +10 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or input Current limit is sufficient to protect the inputs

Tabelle 4-19. Fortsetzung

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			TEMPERATURE
	MIN	TYP	MAX	
T54LS90X T54LS92X T54LS93X	4 5 V	5 0 V	5 5 V	−55°C to +125°C
T74LS90X T74LS92X T74LS93X	4 75 V	5 0 V	5 25 V	0°C to +75°C

X = package type. D for Ceramic Dip, B for Plastic Dip See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2 0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0 7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0 8		
V_{CD}	Input Clamp Diode Voltage			−0 65	−1 5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2 5	3 4		V	V_{CC} = MIN, I_{OH} = −400 μA V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2 7	3 4			
V_{OL}	Output LOW Voltage	54, 74		0 25	0 4	V	I_{OL} = 4.0 mA; V_{CC} = MIN, V_{IN} = V_{IH} or
		74		0 35	0.5	V	I_{OL} = 8.0 mA; V_{IL} per Truth Table
I_{IH}	Input HIGH Current MS, MR $\overline{CP}_0$ $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				 20 120 40 80	μA	V_{CC} = MAX, V_{IN} = 2 7 V
	MS, MR $\overline{CP}_0$, $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				0 1 0 4 0 8	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current MS, MR $\overline{CP}_0$ $\overline{CP}_1$ (LS93) $\overline{CP}_1$ (LS90, LS92)				 −0 4 −2 4 −1 6 −3 2	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 4)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CC}	Power Supply Current			9	15	mA	V_{CC} = MAX

NOTES
1. Conditions for testing, not shown in the table, are chosen to guarantee operation under "worst case" conditions.
2. The specified LIMITS represent the "worst case" value for the parameter. Since these "worst case" values normally occur at the temperature and supply voltage extremes, additional noise immunity and guard banding can be achieved by decreasing the allowable system operating ranges.
3. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C, and maximum loading.
4. Not more than one output should be shorted at a time.

Tabelle 4-19. Fortsetzung

AC CHARACTERISTICS: $T_A = 25°C$, $V_{CC} = 5.0$ V, $C_L = 15$ pF

SYMBOL	PARAMETER	LIMITS LS90 MIN	LS90 MAX	LS92 MIN	LS92 MAX	LS93 MIN	LS93 MAX	UNITS	
f_{MAX}	$\overline{CP}_0$ Input Count Frequency	32		32		32		MHz	Fig 1
f_{MAX}	$\overline{CP}_1$ Input Count Frequency	16		16		16		MHz	Fig 1
t_{PLH} t_{PHL}	Propagation Delay, $\overline{CP}_0$ Input to Q_0 Output		16 18		16 18		16 18	ns	Fig 1
t_{PLH} t_{PHL}	$\overline{CP}_1$ Input to Q_1 Output		16 21		16 21		16 21	ns	
t_{PLH} t_{PHL}	$\overline{CP}_1$ Input to Q_2 Output		32 35		16 21		32 35	ns	
t_{PLH} t_{PHL}	$\overline{CP}_1$ Input to Q_3 Output		32 35		32 35		51 51	ns	
t_{PLH} t_{PHL}	$\overline{CP}_0$ Input to Q_3 Output		48 50		48 50		70 70	ns	
t_{PLH}	MS Input to Q_0 and Q_3 Outputs		30					ns	Fig 3
t_{PHL}	MS Input to Q_1 and Q_2 Outputs		40					ns	Fig 2
t_{PHL}	MR Input to Any Output		40		40		40	ns	Fig 2

AC SET-UP REQUIREMENTS: $T_A = 25°C$, $V_{CC} = 5.0$ V

SYMBOL	PARAMETER	LIMITS LS90 MIN	LS90 MAX	LS92 MIN	LS92 MAX	LS93 MIN	LS93 MAX	UNITS	
t_W	$\overline{CP}_0$ Pulse Width	15		15		15		ns	Fig 1
t_W	$\overline{CP}_1$ Pulse Width	30		30		30		ns	
t_W	MS Pulse Width	15						ns	Fig 2, 3
t_W	MR Pulse Width	15		15		15		ns	Fig 2
t_{rec}	Recovery Time MS to $\overline{CP}$	25						ns	Fig 2, 3
t_{rec}	Recovery Time MR to $\overline{CP}$	25		25		25		ns	Fig 2

RECOVERY TIME (t_{rec}) is defined as the minimum time required between the end of the reset pulse and the clock transition from HIGH-to-LOW in order to recognize and transfer HIGH data to the Q outputs.

AC WAVEFORMS

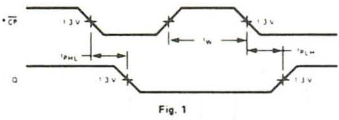


Fig. 1

*The number of Clock Pulses required between the t_{PHL} and t_{PLH} measurements can be determined from the appropriate Truth Tables.

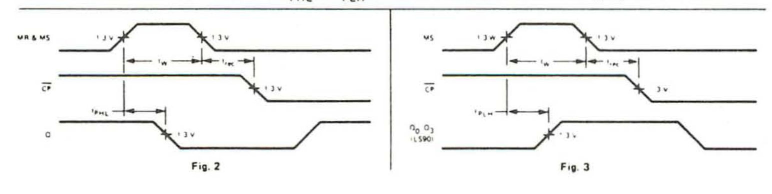


Fig. 2 Fig. 3

Tabelle 4-20. Eigenschaften des 74LS175

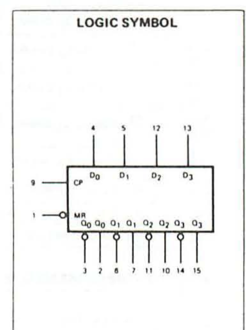
QUAD D FLIP-FLOP

DESCRIPTION – The LSTTL/MSI T54LS175/T74LS175 is a high speed Quad D Flip-Flop. The device is useful for general flip-flop requirements where clock and clear inputs are common. The information on the D inputs is stored during the LOW to HIGH clock transition. Both true and complemented outputs of each flip-flop are provided. A Master Reset input resets all flip-flops, independent of the Clock or D inputs, when LOW.

The LS175 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all SGS-ATES TTL families.

- EDGE-TRIGGERED D-TYPE INPUTS
- BUFFERED-POSITIVE EDGE-TRIGGERED CLOCK
- CLOCK TO OUTPUT DELAYS OF 14 ns
- ASYNCHRONOUS COMMON RESET
- TRUE AND COMPLEMENT OUTPUT
- INPUT CLAMP DIODES LIMIT HIGH SPEED TERMINATION EFFECTS
- FULLY TTL AND CMOS COMPATIBLE

LOGIC SYMBOL

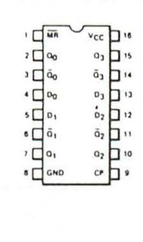


V_{CC} = Pin 16
GND = Pin 8

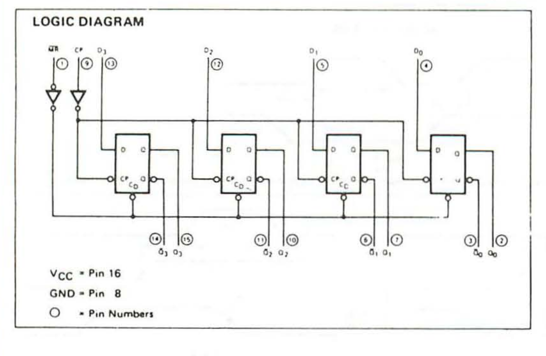
PIN NAMES		LOADING (Note a) HIGH	LOW
$D_0 - D_3$	Data Inputs	0.5 U.L.	0.25 U.L.
CP	Clock (Active HIGH Going Edge) Input	0.5 U.L.	0.25 U.L.
$\overline{MR}$	Master Reset (Active LOW) Input	0.5 U.L.	0.25 U.L.
$Q_0 - Q_3$	True Outputs (Note b)	10 U.L.	5(2.5) U.L.
$\overline{Q}_0 - \overline{Q}_3$	Complemented Outputs (Note b)	10 U.L.	5(2.5) U.L.

NOTES:
a. 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

CONNECTION DIAGRAM
DIP (TOP VIEW)



LOGIC DIAGRAM



V_{CC} = Pin 16
GND = Pin 8
○ = Pin Numbers

Tabelle 4-20. Fortsetzung

FUNCTIONAL DESCRIPTION — The LS175 consists of four edge-triggered D flip-flops with individual D inputs and Q and $\overline{Q}$ outputs. The Clock and Master Reset are common. The four flip-flops will store the state of their individual D inputs on the LOW to HIGH Clock (CP) transition, causing individual Q and $\overline{Q}$ outputs to follow. A LOW input on the Master Reset ($\overline{MR}$) will force all Q outputs LOW and $\overline{Q}$ outputs HIGH independent of Clock or Data inputs.

The LS175 is useful for general logic applications where a common Master Reset and Clock are acceptable.

TRUTH TABLE

Inputs (t = n, $\overline{MR}$ = H)	Outputs (t = n+1) Note 1	
D	Q	$\overline{Q}$
L	L	H
H	H	L

Note 1: t = n + 1 indicates conditions after next clock.

ABSOLUTE MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	−65°C to +150°C
Temperature (Ambient) Under Bias	−55°C to +125°C
V_{CC} Pin Potential to Ground Pin	−0.5 V to +7.0 V
*Input Voltage (dc)	−0.5 V to +15 V
*Input Current (dc)	−30 mA to +5.0 mA
Voltage Applied to Outputs (Output HIGH)	−0.5 V to +10 V
Output Current (dc) (Output LOW)	+50 mA

*Either Input Voltage limit or Input Current limit is sufficient to protect the inputs.

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			TEMPERATURE
	MIN	TYP	MAX	
T54LS175X	4 5 V	5 0 V	5 5 V	−55°C to +125°C
T74LS175X	4 75 V	5 0 V	5 25 V	0°C to +75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2 0			V	Guaranteed Input HIGH Threshold Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0 7	V	Guaranteed Input LOW Threshold Voltage for All Inputs	
		74			0 8			
V_{CD}	Input Clamp Diode Voltage			−0 65	−1 5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2 5	3 4		V	V_{CC} = MIN, I_{OH} = −400 µA	
		74	2 7	3 4			V_{IN} = V_{IH} or V_{IL} per Truth Table	
V_{OL}	Output LOW Voltage	54,74		0 25	0 4	V	I_{OL} = 4 0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or
		74		0 35	0 5	V	I_{OL} = 8 0 mA	V_{IL} per Truth Table
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2 7 V	
					0 1	mA	V_{CC} = MAX, V_{IN} = 10 V	
I_{IL}	Input LOW Current				−0 36	mA	V_{CC} = MAX, V_{IN} = 0 4 V	
I_{SC}	Output Short Circuit Current (Note 4)		−15		100	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current			11	18	mA	V_{CC} = MAX	

Tabelle 4-20. Fortsetzung

NOTES:
1. Conditions for testing, not shown in the Table, are chosen to guarantee operation under "worst case" conditions.
2. The specified LIMITS represent the "worst case" value for the parameter. Since these "worst case" values normally occur at the temperature and supply voltage extremes, additional noise immunity and guard banding can be achieved by decreasing the allowable system operating ranges.
3. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C, and maximum loading.
4. Not more than one output should be shorted at a time.

AC CHARACTERISTICS: T_A = 25°C

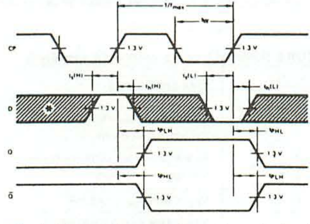
SYMBOL	PARAMETER	LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS
t_{PLH} t_{PHL}	Propagation Delay, Clock to Output		12 15	20 22	ns	Fig. 1
t_{PHL}	Propagation Delay, $\overline{MR}$ to Q Output		20	28	ns	Fig. 2
t_{PLH}	Propagation Delay, $\overline{MR}$ to $\overline{Q}$ Output		16	24	ns	Fig. 2
f_{MAX}	Maximum Input Clock Frequency	40	55		MHz	Fig. 1

AC SET-UP REQUIREMENTS: T_A = 25°C

SYMBOL	PARAMETER	LIMITS MIN	TYP	MAX	UNITS	TEST CONDITIONS
t_WCP	Minimum Clock Pulse Width	15	10		ns	Fig. 1
t_s	Set-up Time, Data to Clock (HIGH or LOW)	10			ns	Fig. 1
t_h	Hold Time, Data to Clock (HIGH or LOW)	0			ns	Fig. 1
t_{rec}	Recovery Time for $\overline{MR}$	12	8.0		ns	Fig. 2
$t_W\overline{MR}$	Minimum $\overline{MR}$ Pulse Width	12	8.0		ns	Fig. 2

AC WAVEFORMS

CLOCK TO OUTPUT DELAYS, CLOCK PULSE WIDTH, FREQUENCY, SET-UP AND HOLD TIMES DATA TO CLOCK



*The shaded areas indicate when the input is permitted to change for predictable output performance.

Fig. 1

MASTER RESET TO OUTPUT DELAY, MASTER RESET PULSE WIDTH, AND MASTER RESET RECOVERY TIME

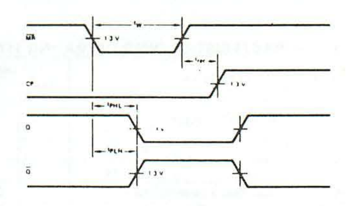


Fig. 2

DEFINITIONS OF TERMS:

SET-UP TIME (t_s) – is defined as the minimum time required for the correct logic level to be present at the logic input prior to the clock transition from LOW to HIGH in order to be recognized and transferred to the outputs.

HOLD TIME (t_h) – is defined as the minimum time following the clock transition from LOW to HIGH that the logic level must be maintained at the input in order to ensure continued recognition. A negative HOLD TIME indicates that the correct logic level may be released prior to the clock transition from LOW to HIGH and still be recognized.

RECOVERY TIME (t_{rec}) – is defined as the minimum time required between the end of the reset pulse and the clock transition from LOW to HIGH in order to recognize and transfer HIGH Data to the Q outputs.

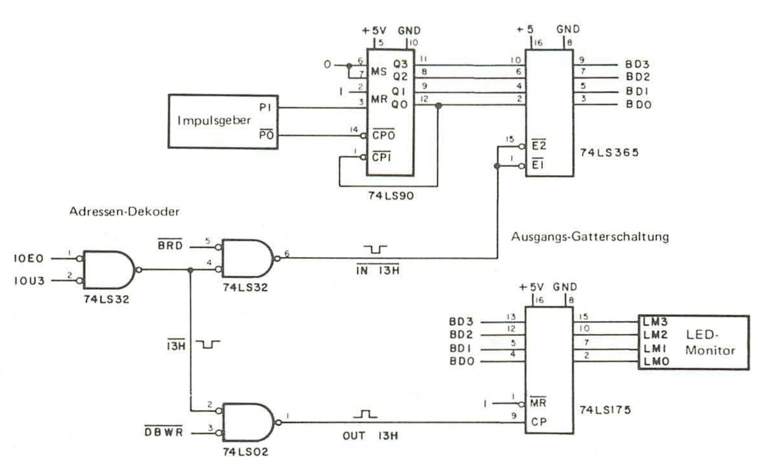


Bild 4-33.

1. Schritt

Bauen Sie die in Bild 4-33 gezeigte Schaltung auf. Betätigen Sie den Impulsgeber P1, damit das IC 74LS90 auf Null gesetzt wird. Der Impulsgeber P1 muß anschließend in seiner Ruhestellung bleiben, damit der Reseteingang Pin 3 logisch 0 bleibt. Anderenfalls kann das IC die Taktimpulse nicht zählen. Laden Sie das Programm UCINM in de Nanocomputer® und führen es aus. Was zeigen die LEDs LM0 . . . LM3 an?
Die LEDs leuchten nicht, d.h. sie dekodieren 0000. Beim Programmablauf erzeugt die CPU mit dem Befehl IN B,(C) den negativen Geräte-Auswahlimpuls $\overline{\text{IN 13H}}$. Dieser Impuls gibt die vier Puffer im IC 74LS365 frei, so daß die CPU die Ausgangsdaten des 74LS90 auf den Daten-BUS legen kann. Da im genannten Beispiel der $\overline{\text{CP0}}$-Eingang noch keinen Taktimpuls erhielt, ist der Ausgangswert 0000. Diesen Wert liest die CPU ins B-Register ein. Der nächste Programmbefehl lautet OUT (C),B, der den positiven Geräte-Auswahlimpuls OUT 13H zur Folge hat. Dieser Impuls tastet ein 4-Bit Auffang-Register in dem Augenblick ab, wo sich die Daten des B-Registers auf den Leitungen BD0 . . . BD3 des Daten-BUS befinden. Da BD0 . . . BD3 mit den D-Eingängen des Flipflops 74LS175 verbunden sind, werden die Signale an die Q-Ausgänge übertragen, wenn der Impuls OUT 13H die Flipflops abtastet. Da die LED-Monitoren LM0 . . . LM3 an den Q-Ausgängen angeschlossen sind, zeigen sie die vom 74LS90 gezählten Impulse an. Die nächste Anweisung ist eine Subroutine CALL zu einer Verzögerungsschleife, welche die CPU ausführt, bevor sie den Vorgang über die Schleife wiederholt.

2. Schritt

Beginnen Sie den Impulsgeber $\overline{\text{P0}}$ schnell zu schalten, während das Pro-

gramm noch läuft. Was beobachten Sie?
Die LED-Monitoren werden regelmäßig auf den neuesten Stand gebracht. Da ein einziger Dezimalzähler nur von 0 . . . 9 zählen kann, entspricht die angezeigte Binärzahl nur der *Einerstelle der gesamt abgetasteten Impulszahl.*

3. Schritt

Das Programm und die Schaltung aus Bild 4-33 machen zwei parallelablaufende Prozesse deutlich. Zum ersten ist es ein Zählvorgang. Von einem Impulsgeber ausgehende Signale werden gezählt und kontrolliert. Der zweite Prozess ist eine ständig von der CPU wiederholte Ein- und Ausgabe-Warteoperation.
Warteschleifen übernehmen in der Regel bei Mikroprozessor-Anwendungen eine wichtige Aufgabe. So könnte im genannten Beispiel die CPU aus dem abgelesenen Zählerstand für das ablaufende Programm wichtige Schlüsse ziehen. So wären insbesondere die Berechnungen der Impuls-Taktfrequenz und die Änderungsgeschwindigkeit der Taktfrequenz möglich. Der kritische Punkt in der Schaltung ist die Unabhängigkeit zwischen Impulszähler und CPU; dennoch muß sie jeden Zählvorgang registrieren. Die Puffer-Auffang-Register gewährleisten die erforderliche Unabhängigkeit. Die CPU kann das Zählerergebnis jederzeit "im Auge behalten" und sich über den Zählerstand informieren. Die Register sind also äußerst nützlich und entsprechend gebräuchlich.
Das nächste Kapitel bespricht die I/O-Hardware und Software; wie der Nanocomputer® sie für die Funktion der Tastenfeld-, Siebensegment- und LED-Anzeigen benutzt. Wenn Sie einmal die Harde- und Software des Nanocomputers® beherrschen, werden Sie auf Ausgabeanzeigen wie im letzten Versuch verzichten. Trotzdem ist die Kenntnis um so einfache Ausgang-Auffang-Register häufig sehr nützlich.

Kapitel 5

Hardware und System-Software

Wie man mit dem Z-80-IC Verbindung aufnimmt, ist Inhalt der ersten Kapitel. Es ist klar, daß die CPU nur ein Teil eines Systems ist, in dem eine Vielzahl von Bauelementen harmonisch und synchron zusammenwirken. So ist es denn auch nicht verwunderlich, daß die Kosten für die CPU gegenüber den Kosten für die anderen Bauelemente relativ gering sind. In diesem Kapitel wird das Z-80-Interfacing durch eine Fallstudie am Nanocomputer® erweitert. Dabei ist die Eingabeeinheit (das Tastenfeld) sowie das Display (die Anzeigeeinheit) von besonderem Interesse. Wie sind beide Einheiten konzipiert und welche Software ist dafür erforderlich? Zwei Fragen, die das Kapitel beantwortet. In einigen Versuchen wird Ihnen deutlich, wie Sie das Tastenfeld und die Anzeige für die eigenen Zwecke nutzen können.

Am Ende dieses Kapitels werden Sie zu folgendem in der Lage sein:

- den Gesamtaufbau der Nanocomputer®-Hardware zu verstehen;
- den Gesamtaufbau der Nanocomputer®-System-Software zu verstehen;
- die Schaltbilder für die Anzeigeeinheit des Nanocomputers® zu lesen und zu deuten;
- die Betriebssystem-Routine für die Anzeigeeinheit im EPROM zu lokalisieren;
- die Anzeige-Treibsubroutine im Betriebssystem des Nanocomputers® einzusetzen, um Anzeigen für Ihre eigenen Zwecke zu produzieren;
- das Schaltbild für das Tastenfeld des Nanocomputers® zu lesen und zu verstehen;
- die Betriebssystem-Routine für das Tastenfeld im EPROM zu lokalisieren;
- die Tastenfeld-Eingaberoutine zur Bestimmung der eigenen Tastenfeldfunktion/Tastenverhältnisse einzusetzen;
- die Serien-I/O-Routine des Nanocomputers® zu lesen und zu verstehen, die zur Kombination mit Seriengeräten z.B. Kassettenrecoder, Tonbänder und ASCII-Anschlüsse benutzt werden.

ÜBERBLICK ÜBER DIE NANOCOMPUTER® -HARDWARE

Bild 5-1 ist ein funktionelles Blockschaltbild der Nanocomputer®-PC-Platine. Bild 5-2 zeigt den mechanischen Aufbau der PC-Platine. Die den verschiedenen Funktionen zugeordneten Bauelemente sind beschriftet. Das Serial-Interface (serielle Schnittstelle) und der DC/DC-Wandler (DC = Gleichstrom) sind nicht Bestandteil des Standard-Nanocomputers®, man kann sie jedoch als Ergänzungsbausatz nachträglich einbauen.

In den ersten 4 Kapiteln dieses Buches sind viele Funktionen des Nanocomputers® erörtert. In Kapitel 1 ist die Z-80-CPU detailliert behandelt. Das Dekodieren von Speicheradressen sowohl für RAM und EPROM als auch für die I/O-Gatter-Auswahl für den Nanocomputer® beschreibt Kapitel 3. Im Kapitel 4 sind Techniken für BUS-Anschlüsse und Mikrocomputer-I/O definiert. Die noch verbleibenden Themen sind:

- Der Nanocomputer®-BUS, "Erweiterungs-BUS" genannt, der die Erweiterungs-BUS-Treiber sowie die Anschlüsse J1 und J2 mit einschließt;
- die magnetischen Band- und Serieneinheiten mit ihren Anschlüssen J3a, J3b und J5;
- die vier I/O-Gatter in Form von zwei Z-80-PIO-ICs mit den Anschlüssen J6 und J7.

Das vorliegende Kapitel behandelt die beiden ersten Themen, während auf das PIO-IC Kapitel 7 näher eingeht.

Der Nanocomputer®-BUS

Die Bilder 5-3 und 5-4 zeigen die Treiberschaltung für den Nanocomputer®-BUS. Die für den Nanocomputer® in Frage kommenden elektrischen Anschlüsse können in fünf Kategorien eingeteilt werden:

KATEGORIE 1: Mit Three-State-Puffer/Treiber verbundene Leitungen
Jedes Signal auf dem Nanocomputer®-BUS, das über eine Leitung der Kategorie 1 geht, wird von einem Three-State-Puffer/Treiber getrieben. Der Puffer nimmt einen maximalen Strom von 24 mA (Ausgangsspannung 0,5 V) und kann bei einer Versorgungsspannung von +5 V einen Strom von 2,6 mA bei einer Ausgangsspannung von >2,4 V liefern. Die Treiberausgänge sind hochohmig, wenn das Signal $\overline{\text{BUSAK}}$ logisch 1 ist. Leitungen der ersten Kategorie niemals von mehr als einem Treiber gleichzeitig treiben lassen.

KATEGORIE 2: Bidirektionale Leitungen
Die Leitungstreiber der Kategorie 2 ähneln denen der Kategorie 1. Angeschlossene Empfänger sind in der Regel ICs der Serie 74LSXXX. Der Ausgangsstrom beträgt unter 0,4 V (low) 0,36 mA und im High-Zustand (über 2,7 V) 20 mA. Die bidirektionalen Leitungen auf dem Nanocomputer®-BUS werden von den Signalen $\overline{\text{DBIN}}$ und $\overline{\text{DBOUT}}$ gesteuert und sind daher niemals gleichzeitig aktiv.

KATEGORIE 3: Leitungen, die mit der internen CPU des Nanocomputers® verbunden sind
Bestimmte Steuersignale können nur von der internen CPU auf der Platine des Nanocomputers® getrieben werden; zusätzliche externe CPUs haben darauf keinen Einfluß. (Ein interessantes Thema sind BUS-Leitungen für mehrere CPUs. Die Themenbehandlung würde jedoch über den Rahmen dieses Buches hinausgehen.) Es

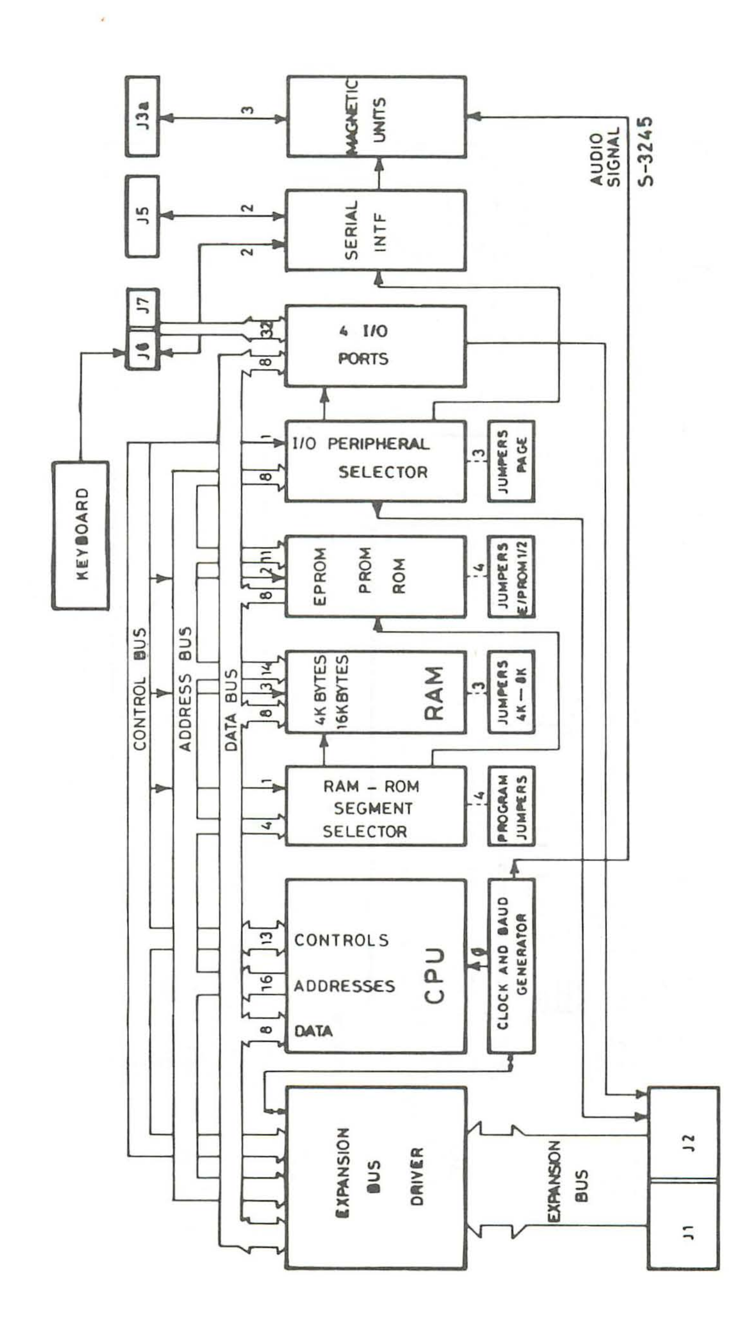


Bild 5-1. Funktionelles Blockschaltbild einer Nanocomputer®-PC-Platine. Die einzelnen Funktionsblöcke sind mit den englischen Bezeichnungen versehen.

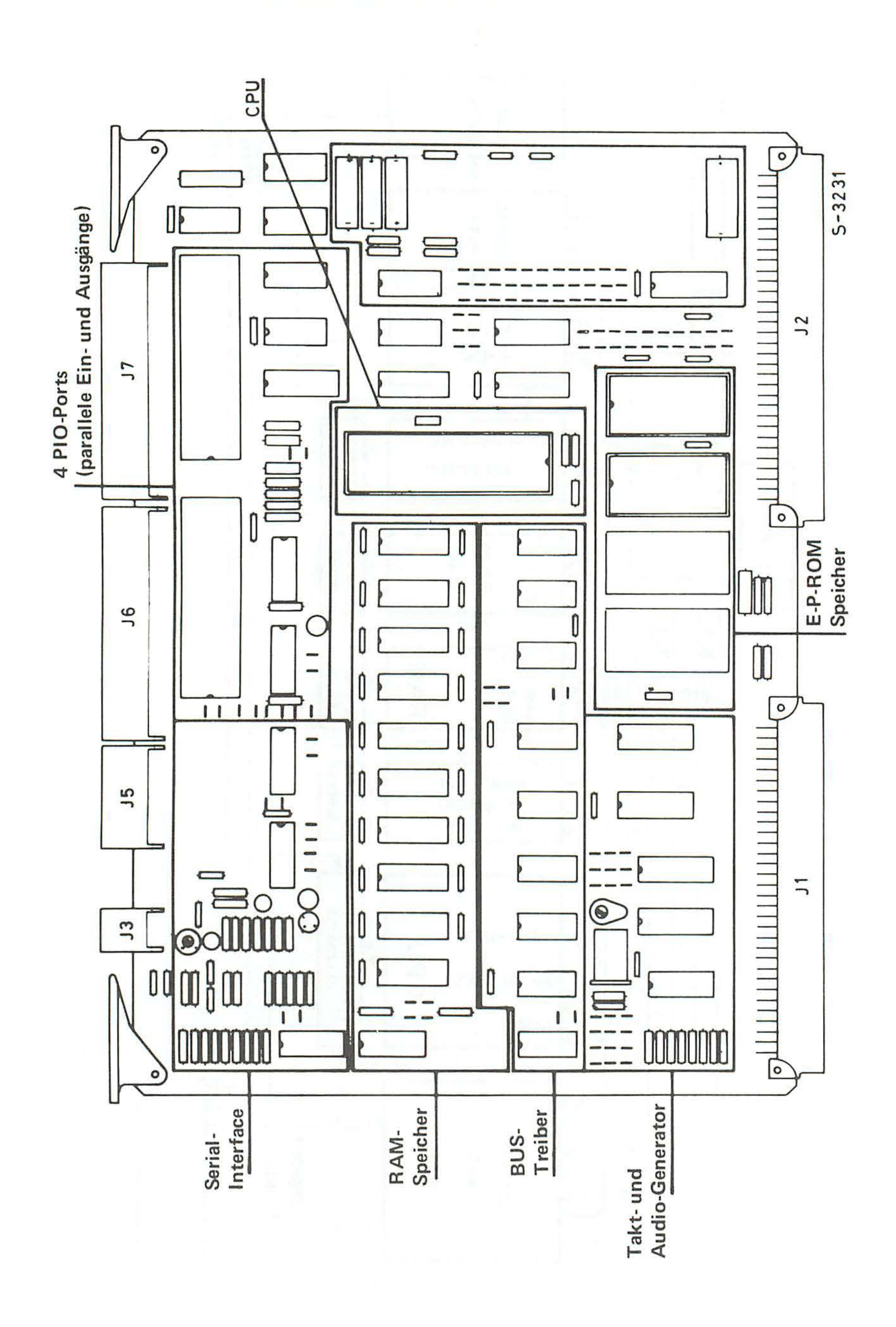


Bild 5-2. Anordnung der Bauelemente auf der Nanocomputer®-PC-Platine.

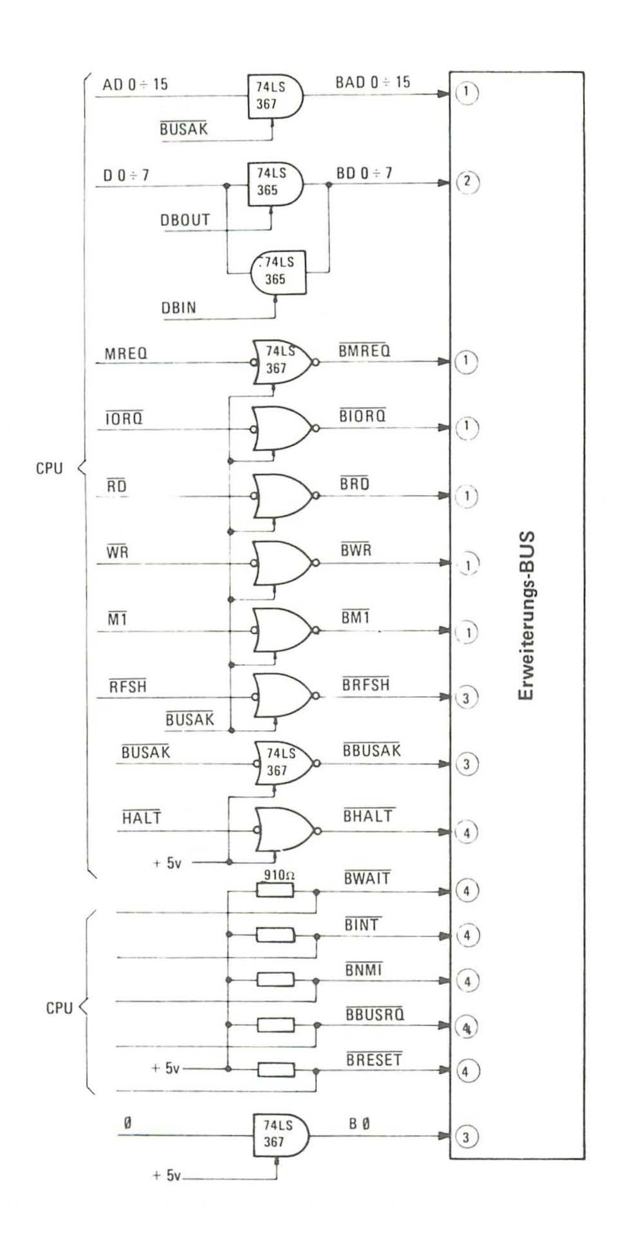


Bild 5-3. Das Schaltbild zeigt die Anschlüsse beim Erweiterungs-BUS des Nanocomputers®.

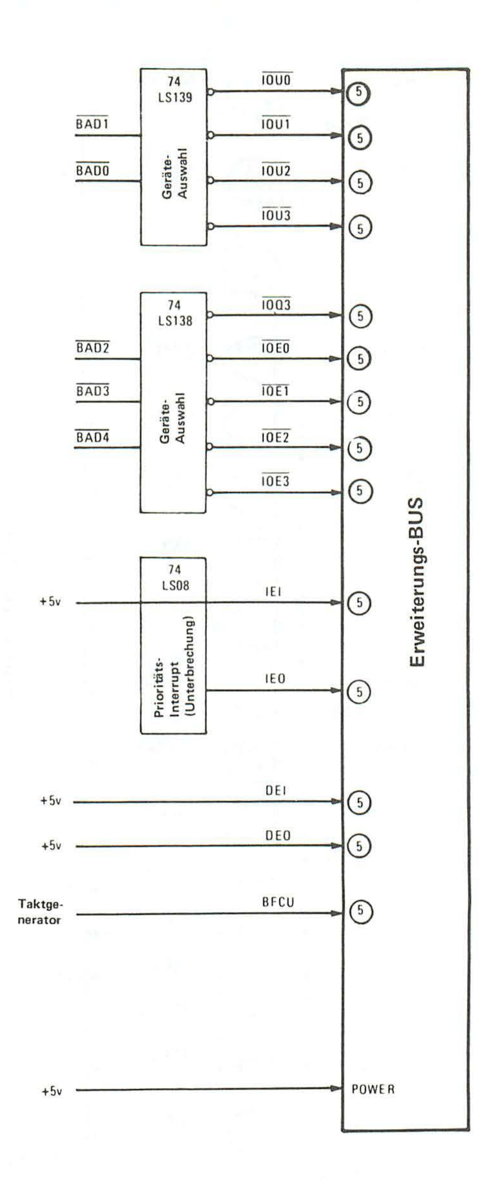


Bild 5-4. Das Bild zeigt die noch fehlenden Anschlüsse zum Erweiterungs-BUS.

ist durchaus denkbar, daß der Nanocomputer® noch mit einer weiteren CPU korrespondiert. Beide CPUs teilen sich dann den vorhandenen BUS, den Speicher und die I/O-Gatter. Mit anderen Worten kann eine externe CPU den BUS des Nanocomputers® benutzen

sowie I/O- bzw. Speicher-Lese- und -Schreibsignale ausgeben, um zu den externen Geräten und dem Speicher des Nanocomputers® Zugang zu finden. Eine begrenzte Zahl von Signalen können von der CPU des Nanocomputers® nur intern und nicht von anderen Geräten auf dem BUS getrieben werden, da solche Geräte lediglich Benutzer solcher Leitungen sind. Letztere gehören der Kategorie 3 an und führen nur die Signale $\overline{\text{RFSH}}$, Systemtakt und $\overline{\text{BUSAK}}$. Die genannten Leitungen arbeiten mit Puffer/Treiber ähnlich wie in Kategorie 1; allerdings dürfen sie niemals einen hochohmigen Zustand annehmen.

KATEGORIE 4: *Leitungen, die an offene Kollektorausgänge angeschlossen sind*
Mit offenen Kollektorausgängen verbundene Leitungen sind zu einem "Wired-OR" zusammengeschaltet. Ist nur ein Ausgangssignal logisch 0, ist die mit der "Wired-OR"-Schaltung verbundene BUS-Leitung ebenfalls logisch 0. Jedes IC mit offenem Kollektorausgang, das mehr als 8 mA bei einer Spannung unter 0,5 V verbrauchen kann, ist dazu geeignet, die externen Signale am BUS zusammenzuschließen als Eingang für eine Leitung der Kategorie 4. Beachten Sie die Pull-up-Widerstände an den Eingängen der Kategorie 4 in Bild 5-3.

KATEGORIE 5: *Service-, Prioritäts-, Interrupt- und Zugriffsleitungen*
Diese Leitungen transportieren vom Nanocomputer® erzeugte Signale, die von anderen ICs auf dem BUS benutzt werden können. Jede Leitung kann bis zu 10 Empfänger-ICs der gleichen Serie bedienen. Ist eine höhere Ladung erforderlich, muß man ICs mit höheren Lastfaktoren verwenden.

In Tabelle 5-1 sind die Bezeichnungen, Funktionen, Kategorien sowie die Pin-Zahlen der Anschlüsse J1 und J2 aller BUS-Signale des Nanocomputers® zusammengestellt.

Kassettenanschluß sowie serieller Ein- und Ausgang

Der Nanocomputer® kann maximal zwei Kassettenrekorder als Bandspeicher bedienen. Dafür sind die Anschlüsse J3a und J3b auf der PC-Platine vorgesehen. Die Eingangsempfindlichkeit der Kassettenrekorder muß zwischen 10 mV und 50 mV betragen. Für die Anpassung des Eingangspegel ist das Trimmpotentiometer R54 auf der PC-Platine vorgesehen. Notfalls muß noch ein externes Widerstandsnetzwerk hinzu geschaltet werden. Der Kassettenausgangspegel sollte zwischen 300 und 400 mV liegen.

Das Kassetten-Interface steuert über den REMOTE CONTROL Pin (Fernsteuerungsanschluß) den Bandlauf beim Rekorder. Einzelheiten über den Aufbau des Interface folgen in einem späteren Abschnitt dieses Kapitels. Dort sind auch die LD-(LOAD-)- und DP-(DUMP-)-Routinen beschrieben.

Tabelle 5-1. Belegung der Anschlußleisten J1 und J2 beim Erweiterungs-BUS des Nanocomputers®.

Bezeichnung	Funktion	Kateg.	Anschluß
+ 5V	5V Supply ± 2%	—	J1-1ac/J2-1ac
$\overline{\mathrm{BMREQ}}$	Memory Request	1	J2-22c
$\overline{\mathrm{BIORQ}}$	Input-Output Request	1	J2-21c
$\overline{\mathrm{BRD}}$	Read from memory Data or Peripheral	1	J2-13c
$\overline{\mathrm{BWR}}$	Write to memory or peripheral	1	J2-14c
$\overline{\mathrm{BM1}}$	Machine Cycle 1	1	J1-15c
$\overline{\mathrm{BRFSH}}$	Refresh Cycle	1	J1-11c
$\overline{\mathrm{BBUSAK}}$	Bus Acknowledge	3	J1-20c
$\overline{\mathrm{BBUSRQ}}$	Bus Request	4	J2-25c
$\overline{\mathrm{BHALT}}$	Halt	3	J1-18c
$\overline{\mathrm{BWAIT}}$	Wait	4	J2-26c
$\overline{\mathrm{BINT}}$	Interrupt Request	4	J2-24c
$\overline{\mathrm{BNMI}}$	Nonmaskable Interrupt Request	4	J2-23c
$\overline{\mathrm{BRESET}}$	Reset	4	J1-28c
B0	Machine Clock	3	J1-17c
$\overline{\mathrm{BFCU}}$	Conversion Clock	5	J1-8c
$\overline{\mathrm{IOU0}}$		5	J1-24c
$\overline{\mathrm{IOU1}}$		5	J1-23c
$\overline{\mathrm{IOU2}}$	Address Decode for lines BAD0, BADL	5	J1-22c
$\overline{\mathrm{IOU3}}$		5	J1-21c
$\overline{\mathrm{IOQ3}}$		5	J2-5c
$\overline{\mathrm{IOE0}}$	Partial Address Decode (3, 4, 5, 6, 7)	5	J2-6c
$\overline{\mathrm{IOE1}}$	for lines BAD2, BAD3, BAD4.	5	J2-8c
$\overline{\mathrm{IOE2}}$		5	J2-9c
$\overline{\mathrm{IOE3}}$		5	J2-11c
+ 12V		—	J1-16ac/J2-16ac
− 12V			J2-3ac
− 5V			J2-4ac
GND	Power Supply return	—	J1-32ac/J2-32ac
BD0	Least significant bit	2	J1-27c
BD1		2	J1-26c
BD2		2	J1-25c
BD3		2	J1-29c
BD4	Data Lines	2	J1-30c
BD5		2	J1-31c
BD6		2	J2-12c
BD7	Most significant bit	2	J2-10c
BA0	Least significant bit	1	J1-3c
BA1		1	J1-7c
BA2		1	J1-6c
BA3	Address Lines	1	J1-3c
BA4		1	J1-4c
BA5		1	J1-5c
BA6		1	J2-30c
BA7		1	J2-29c
BA8		1	J2-31c
BA9		1	J2-28c
BA10		1	J1-12c
BA11	Address Lines	1	J2-27c
BA12		1	J2-19c
BA13		1	J2-7c
BA14		1	J2-18c
BA15	Most significant bit	1	J2-17c

Anmerkung: Die BUS-Leitungen BAD0 . . . BAD15 sind mit den Leitungen BA0 . . . BA15 der Experimentierplatine identisch.

In der nachfolgenden Auflistung sind die englischen Bezeichnungen verdeutscht:

5 V-Versorgungsspannung ± 2%
Speicher-Abruf
Eingangs-/Ausgangs-Abruf

Ablesen der Daten vom Speicher oder einem Peripheriegerät
Einschreiben der Daten in den Speicher oder in ein Peripheriegerät

Maschinenzyklus 1
Refresh-Zyklus (Speicherauffrischung)
BUS-Bestätigung
BUS-Abruf
HALT-Befehl
Wartesignal
Interrupt-Abruf
Nichtmaskierbarer Interrupt-Abruf
Rücksetzen
Maschinentakt
Konversionstakt
Adressen-Dekoder der Leitungen BADO, BAD1
Bevorzugte Adressendekodierung (3, 4, 5, 6, 7) der BUS-Leitungen BAD2, BAD3, BAD4
Masse
Niederwertiges Bit
Daten-Leitungen
Höchstwertiges Bit
Niederwertiges Bit
Adressen-Leitungen
Adressen-Leitungen
Höchstwertiges Bit

Die DP-Routine zeichnet Daten auf die Kassette auf, während die LD-Routine Daten von der Kassette in den Speicher des Nanocomputers® einschreibt. Die Aufschlüsselung des Anschlusses J3 zeigt Tabelle 5-2.
Die serielle Schnittstelle des Nanocomputers® (Serial-Interface) ermöglicht Anschluß von Geräten mit unterschiedlichen Interface-Normen. Die wichtigste und in der Regel immer zutreffende Norm ist die TTL-Kompatibilität. Die Auswahl der Interface-Norm erfolgt auf der PC-Platine mit Hilfe von Steckverbindungen.

ÜBERSICHT ÜBER DIE SYSTEM-SOFTWARE DES NANO-COMPUTERS®

Bild 5-5 zeigt das Flußdiagramm des Nanocomputers®-Operationssystems. Das Operationssystem hat vier verschiedene Eingangspunkte.

Tabelle 5-2. Anschlüsse J3a und J3b

Anschluß	Bezeichnung	Funktion
1	IM1	Signaleingang von Kassette 1
2	IM2	Signaleingang von Kassette 2
3	GND	Masse
4	GND	Masse
5	UM1	Signalausgang zur Kassette 1
6	UM2	Signalausgang zur Kassette 2
7	CA1ON	Ausgang für automatischen Start/Stop der Kassette 1
8	CA2ON	Ausgang für automatischen Start/Stop der Kassette 2

Tabelle 5-3

Anschluß	Bezeichnung	Funktion
1	REAR	Ausgang für Papierstreifen-Transportkontrolle (beim Nanocomputer® unbenutzt)
2	RTXTTY	20 mA-Schleife für serielle TTY (Senderschleife) mit Anschluß 5
3	RTXTTY	wie bei Anschluß 2
4	RREAR	Schleife mit Anschluß 1 (unbenutzt)
5	TXTTY	20 mA-Schleife für serielle TTY (Senderschleife) mit Anschluß 2
6	DTR	Daten-Terminal bereit zur Ausgabe (unbenutzt)
7	DSR	Daten bereit zur Eingabe (unbenutzt)
8	RTS	Abruf der Ausgangsdaten (unbenutzt)
9	SICK	Modulator/Demodulatorausgang für Baudgeschwindigkeit (unbenutzt)
10	GND	
11	GND	
12	+5 V	
13	GND	
14	RRXTTY	20 mA-Schleife für serielle TTY (Empfängerschleife) mit Pin 18
15	TXTTL	TTL-Pegel (Sender)
16	RXTTL	TTL-Pegel (Empfänger)
17	CTS	Fertig zum Senden (unbenutzt)
18	RXTTY	20 mA-Schleife für serielle TTY (Empfängerschleife) mit Pin 14

RESET/STROMVERSORGUNG EIN

Diese Funktion wird ausgeführt, wenn man den Nanocomputer® rücksetzt (durch Betätigen der RESET-Taste) oder die Stromversorgung einschaltet. Mit einem einfachen Speichertest überprüft die CPU die vom Operationssystem für Stapelspeicher und Daten benutzen Stellen im Schreib/Lesespeicher. Nach erfolgreicher Durchführung des RAM-Tests stellt das Operationssystem das Standard-Bandformat und die Baudgeschwindigkeit (600 Baud) ein. Anschließend übernimmt der zwischengespeicherte CPU-Inhalt die Steuerung.

RST 38H

Auch mit diesem Befehl übernimmt das Operationssystem die Steuerung des Nanocomputers®. Den Rücksprung aus einem vom Benutzer eingegebenen Programm zum Operationssystem (Speicherstelle 0038H) haben Sie bereits in Buch 1 kennengelernt. Es werden anschließend keinerlei Aufgaben mehr ausgeführt, bevor man nicht die Schrittfolge einleitet, die mit der Zwischenspeicherung des CPU-Inhaltes beginnt.

NMI

Die dritte Möglichkeit, in das Betriebssystem des Nanocomputers® zurückkehren, ist die nichtmaskierbare Unterbrechung. Zwei Dinge können eine nicht maskierbare Unterbrechung zur CPU auslösen: entweder das Drücken der BREAK-Taste oder die kurzzeitige Erdung des CPU-$\overline{\text{NMI}}$-Eingangs. In Kapitel 6 führen Sie mehrere Versuche mit dem $\overline{\text{NMI}}$-Eingang durch.
Aus Bild 5-5 geht hervor, warum es zwischen dem Drücken der BREAK-Taste und der RESET-Taste einen Unterschied gibt. Das Drücken der RESET-Taste löst zwei zusätzliche Funktionen aus, nämlich den Speichertest und die Einstellung des Bandformats sowie der Baud-Geschwindigkeit. Diese Funktionen stellen dabei die CPU-Register auf Null. Wie bereits erwähnt, sollte man die BREAK-Taste anstelle der RESET-Taste drücken, wenn das Anwenderprogramm die Steuerung an das Operationssystem zurückgeben soll, ohne den CPU-Inhalt zu zerstören.

EINZELSCHRITT

Läuft ein Anwenderprogramm in Einzelschritten ab, kommt immer wieder das Operationssystem zur Anwendung. Dabei führt die CPU abwechselnd den Einzelbefehl und dann mehrere Befehle vom Operationssystem aus, sie sind aus dem Flußdiagramm in Bild 5-5 ersichtlich.
Sobald eine der vier genannten Bedingungen die Steuerung an das Operationssystem übertragen hat, laufen zusätzlich fünf Grundsatz-Funktionen ab.
Zunächst wird der Zustand aller CPU-Register durch Verschieben ihres Inhalts in den System-Stapelspeicher zwischengespeichert. Größe, Speicherstelle und genaue Verwendung der System-Software-Daten und Stapelspeicherbereiche des RAM ist an anderer Stelle dieses Kapitels detailliert beschrieben. Die nächste vom Operationssystem durchgeführte Aufgabe ist die Zusammenstellung und Anzeige der entsprechenden Daten auf dem peripheren Display. Wie Sie beobachten können, wird die Art dieser An-

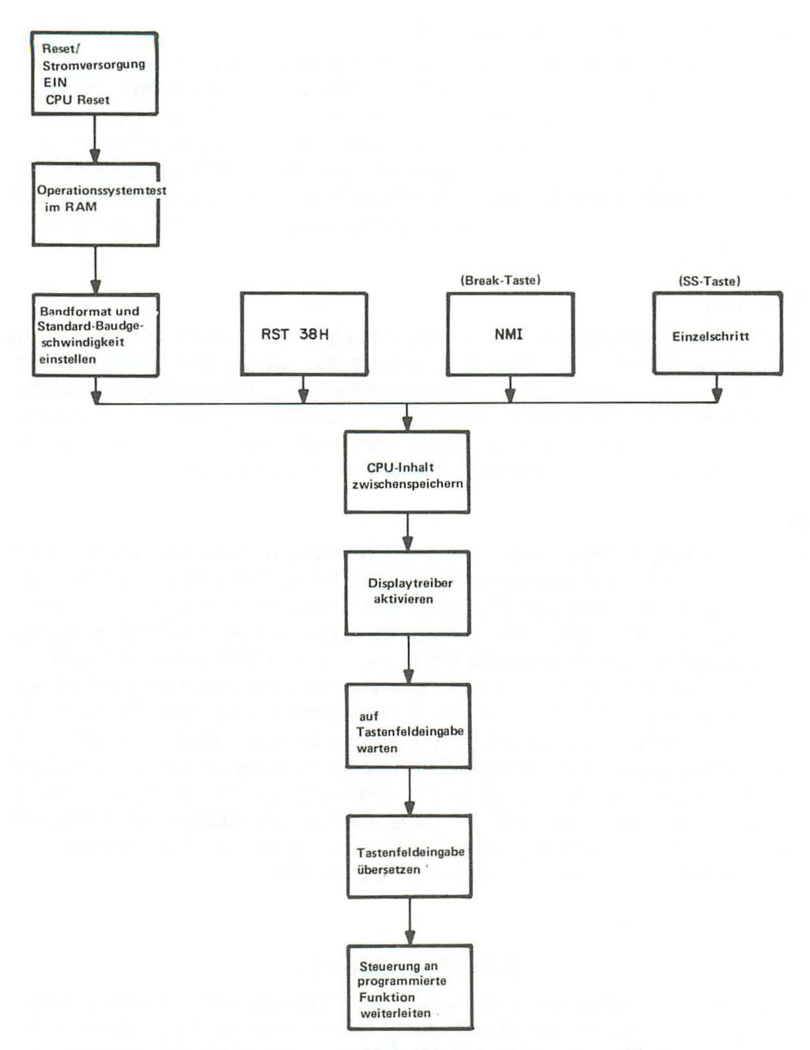


Bild 5-5. Flußdiagramm der System-Software beim Nanocomputer®.

zeige durch die Stellung der Selektor-LED und den Inhalt des oder der anzuzeigenden Register bestimmt. Als nächstes wartet das Operationssystem auf einen Tastendruck durch den Benutzer. Wie Sie aufgrund genauer Beschreibung über den Anzeigentreiber und die Tastenfeld-Eingaberoutine an anderer Stelle in diesem Kapitel feststellen, liest das Operationssystem abwechselnd das Tastenfeld-Eingabegatter (Input-Port) ab und überschreibt den Inhalt in das Anzeige-Ausgangsgatter (Output-Port). Dieses abwechselnde Lesen und Schreiben bringt das Display immer auf den neuesten Stand. Ist ein vollständiger Befehl verstanden worden, geht die Steuerung an die Funktion über, die den Befehl ausführt.

Das restliche Operationssystem besteht aus Software-Modulen. Für jeden möglichen Tastenfeld-Befehl ist ein Modul vorhanden. Deshalb gibt es Module zur Ausführung der Tastenfunktionen GO, DUMP, LD, ST und INC als auch zur Umsetzung aller numerischen Hex-Eingaben 0, 1 . . . F. Schließlich sind noch Module vorhanden, die die z.Zt. aufleuchtenden Anzeigen um eine Stelle weiterschieben und die Zahl der zuletzt gedrückten Zifferntaste anzeigen. Zu jeder Taste auf dem Tastenfeld gehört also ein Software-Modul, das im entsprechenden Fall die Steuerung übernimmt.
Sobald das angesprochene Modul die Steuerung beendet, wird sie durch den Befehl RST 38H in das Operationssystem zurückgeführt.
Der Benutzer hat zwei Möglichkeiten die CPU zu steuern. Die erste ist über den GO-Befehl. Nachdem man die GO-Taste betätigt hat, gibt das Operationssystem die Steuerung zum "GO-Modul", das einen einzigen Schritt durchführt und dann einen Sprung zu der im PC-Register enthaltenen Adresse macht. Die zweite Methode, das Anwenderprogramm ablaufen zu lassen, ist mit der Einzelschritt-Taste möglich. In diesem Fall wechselt die Steuerung wie bereits erwähnt zwischen dem Anwenderprogramm und dem Operationssystem hin und her.

SUBSYSTEM DER NANOCOMPUTER® -ANZEIGE

Hardware

Bild 5-6 ist ein Schaltbild der vom Nanocomputer® für einfache Benutzer/CPU-Kommunikationen angewandten Hardware. Dieser Abschnitt macht Sie mit den Bauelementen bekannt, welche die acht 7-Segmentanzeigen (LD1 und LD2) sowie die 14 LEDs (L1 bis L14) auf der nach Kundenwünschen konzipierten Tastenfeld/Anzeigeeinheit treiben. Sowohl das Tastenfeld und die Anzeige als auch das Software-Operationssystem sind vom Hersteller so konzipiert, daß sie sich gegenseitig ergänzen und dem Benutzer ein Maximum am Flexibilität und Einfachheit im Zusammenwirken mit der Z-80-CPU bieten. Zusätzlich ist auf Wunsch Interfacing für ASCII-Terminals möglich. Bei verbesserten Nanocomputer®-Typen ist die Serien-Kommunikation der Hardware realisiert, während der Standard-Nanocomputer® jedoch Software benutzt (Näheres im Verlauf dieses Kapitels).
Die in Bild 5-6 gezeigte Schaltung beinhaltet die Tastatur- und Display-Hardware: die Eingangsstufe, welche die Tastenfeldsignale zur CPU weiterleitet und die Ausgangsstufe, welche die 7-Segmentanzeigen steuert. Es sind dies im einzelnen folgende wichtige Bauelemente:

- Q1: das IC BGY16 ist ein Transistor-Array mit 7 einzelnen PNP-Transistoren im 16 Pin-Gehäuse;
- Q2 und Q3: Transistor-Array mit je 7 einzelnen NPN-Darlington-Transistoren im 16 Pin-Gehäuse;
- Q4: HCF4514, ein COS/MOS 4-Bit-Register/4 zu 16-Dekoder, der nach Wahl eine logische 1 ausgibt;
- L1-L14: 14 LEDs;
- LD1, LD2: Anordnung von je vier 7-Segment-Anzeigen pro Reihe.

Und auf der Z-80-PC-Platine:

- Z-80-PIO 1: Z-80-PIO- (Parallel-I/O-Interface)-Steuerung, deren Leitungen in Bild 5-6 mit PB0-PB7 bezeichnet sind.

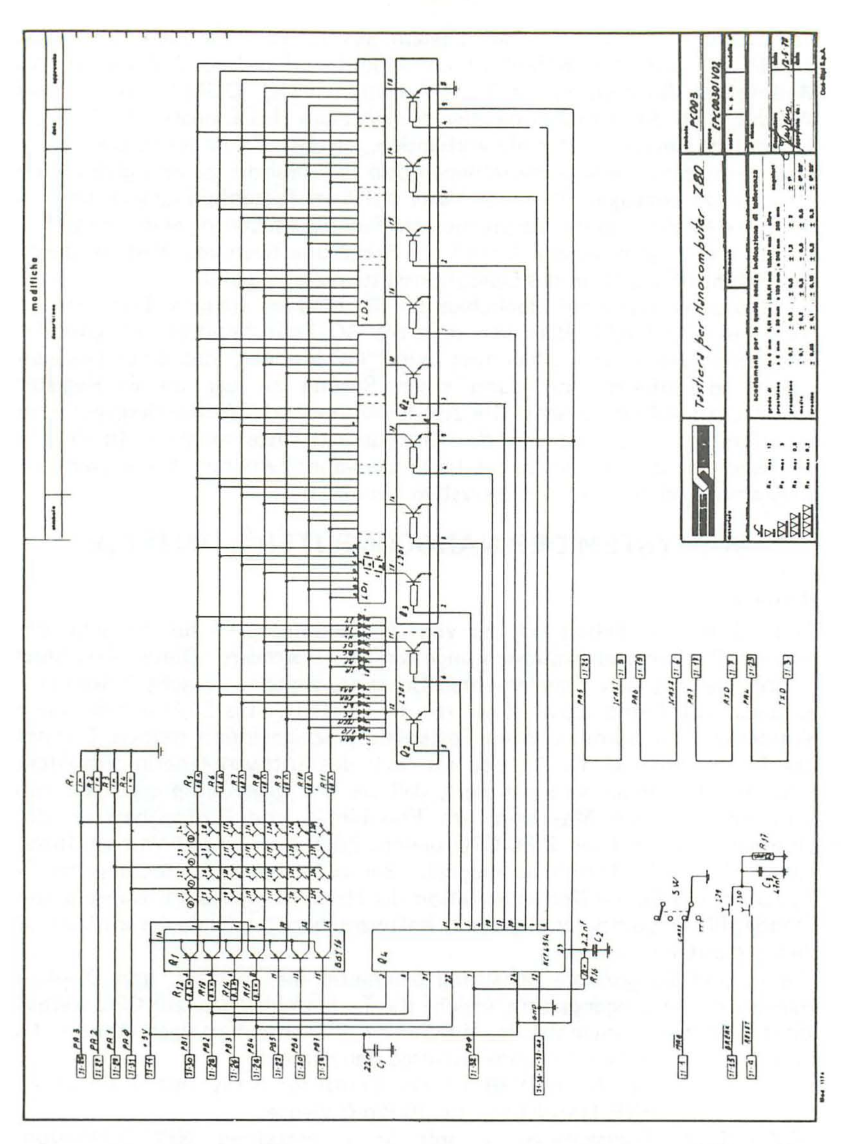


Bild 5-6. Schaltbild der Tastenfeld/Anzeige-Hardware des Nanocomputers®.

Da auf eine langatmige Beschreibung der elektrischen Eigenschaften aller dieser Bauelemente verzichtet werden kann, nachstehend eine Zusammenfassung der wichtigsten Merkmale, die zum Verständnis der Schaltung in Bild 5-6 erforderlich sind.

Dioden- und LED-Anzeigen

Eine Diode ist ein elektronisches Halbleiter-Bauelement, das Strom in nur eine Richtung und nicht in die entgegengesetzte Richtung fließen läßt. Das Symbol für eine Diode ist in Bild 5-7 dargestellt. Der Pfeil ist das typische Zeichen in der Elektronik für das Fließen eines Stroms von einem positiven Potential (+) in die Richtung eines negativen Potentials (–).

Eine wie in Bild 5-8 beschaltete Diode läßt zwar den Strom fließen, wird jedoch schnell zerstört, weil ein strombegrenzender Widerstand fehlt.

Wenn ein definierter Strom über eine Diode fließt, ist die Diode *in Durchlaßrichtung betrieben*. Dagegen fließt bei der in Bild 5-9 dargestellten Situation kein Strom.

Die Anschlußbezeichnungen bei der Diode sind *Anode und Kathode.* In Bild 5-10 sind die beiden Anschlüsse für eine typische Diode dargestellt.

Auf Grund der gemachten Aussagen läßt sich also folgendes feststellen:

- Eine Diode blockiert den Strom, wenn die Anode gegenüber der Kathode negativ ist.
- Eine Diode leitet den Strom, wenn ihre Anode der Kathode gegenüber positiv ist.



Bild 5-7. Symbol einer Halbleiterdiode. Das Symbol A entspricht DIN 40700; in der Literatur findet man jedoch häufig das veraltete Symbol B.

+ 5V ——▶|—— GND

Bild 5-8. In Durchlaßrichtung betriebene Diode.

GND ——▶|—— +5V

Bild 5-9. Nichtleitende Diode.

Anode ——▶|—— Kathode

Bild 5-10. Anode und Kathode für eine Diode.

Eine *Leuchtdiode* oder LED ist eine Diode, die leuchtet, wenn sie in Durchlaßrichtung betrieben wird und Strom durchfließen kann. Das Symbol für eine LED ist in Bild 5-11 dargestellt. Der Anoden- und Kathodenanschluß einer LED sind dieselben wie bei einer normalen Diode (siehe Bild 5-12).

Alle Dioden – einschließlich der LEDs – sind ziemlich empfindliche Halbleiter Bauelemente, die zerstört werden, wenn ein zu starker Strom von der Anode zur Kathode fließt.

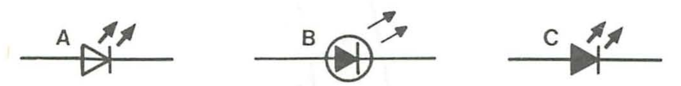


Bild 5-11. Symbol einer Leuchtdiode (LED). Symbol A entspricht der DIN-Vorschrift 40700. In der Literatur sind auch die Symbole B und C üblich, häufig findet man das Symbol C.



Bild 5-12.

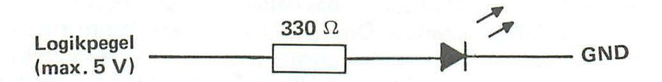


Bild 5-13. Leuchtdiodenschaltung.

Um die LED vor Zerstörung zu schützen, ist ein Vorwiderstand erforderlich. Der Wert muß so bemessen sein, daß der durch die LED fließende Strom auf einen unkritischen Wert begrenzt ist. Gleichzeitig muß jedoch soviel Strom fließen, daß die LED mit genügender Intensität aufleuchtet. Die Widerstände R5 . . . R11 in Bild 5-6 sind alle strombegrenzende Widerstände für die 7-Segment-Anzeigen und die 14 Selektor-LEDs. Ein LED-Schaltungsbeispiel mit einem strombegrenzenden Widerstand von 330 Ohm ist in Bild 5-13 dargestellt.
Die typische rot oder grün leuchtende LED besteht aus einem kleinen Halbleiter-Mikroplättchen (Bild 5-14). Das Bauelement ist mit einem klaren Epoxydharz-Gehäuse umgeben, das dem Auge eine viel größere Fläche vermittelt; der Lichtdurchmesser erhöht sich um ca. das 2½-fache.

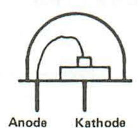


Bild 5-14. Schematischer Aufbau einer LED.

Man kann LEDs auch in anderer Form für andere Anzeigearten herstellen. Die in Bild 5-15 gezeigte 7-Segment-Anzeige ist ein Beispiel für 7 LEDs in Stabform. Zu jeder stabförmigen LED gehört einer der Buchstaben a, b, c, d, e, f oder g. Jede Hex-Ziffer von 0 . . . F läßt sich mit den sieben Segmenten darstellen. Entsprechend der Ziffer leuchten bestimmte Segmente auf. Die folgende Aufstellung zeigt den Zusammenhang zwischen Hex-Ziffer und Segmentkombination.

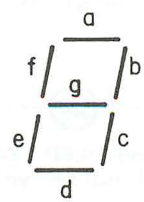


Bild 5-15. 7-Segment-Anzeige.

Hex-Ziffer	aufleuchtende Segment-kombination
0	a, b, c, d, e, f
1	b, c
2	a, b, d, e, g
3	a, b, c, d, g
4	b, c, f, g
5	a, c, d, f, g
6	a, c, d, e, f, g
7	a, b, c
8	a, b, c, d, e, f, g
9	a, b, c, f, g
A	a, b, c, e, f, g
b	c, d, e, f, g
C	a, d, e, f
d	b, c, d, e, g
E	a, d, e, f, g
F	a, e, f, g

Wie bereits erwähnt, muß die Anode einer LED mit +5 V und die Kathode mit Masse verbunden sein, wenn die LED aufleuchten soll. Dabei muß der Strom auf ein zulässiges Maß begrenzt werden. Im Schaltbild 5-6 sind die Selektor-LEDs L1 . . . L14 nicht direkt mit der Versorgungsspannung verbunden, denn es darf ja nur die jeweils angesteuerte LED aufleuchten. Dafür sorgt einmal das Transistor-Array BGY16 (Q1) und zum anderen der Dekoder HCF4514 (Q4) mit zwei Transistoren des Arrays Q2. Soll eine der 14 LEDs aufleuchten, müssen zwei Bedingungen erfüllt sein. Es muß ein Kollektor eines PNP-Transistors im IC Q1 logisch 1 und ein Kollektor der zwei NPN-Darlington-Transistoren im IC Q2 logisch 0 sein. In ähnlicher Weise werden die 7-Segmentanzeigen LD1 und LD2 angesteuert. Auch hierbei müssen die entsprechenden Transistoren in Q1 am Kollektor logisch 1 und die NPN-Transistoren in Q1 und Q2 am Kollektor logisch 0 sein. Die Basis der PNP-Transistoren im IC Q1 sind mit dem B-Gatter im PIO 1 verbunden. Dadurch nimmt die Information auf dem Daten-BUS Einfluß darauf, welche LEDs bzw. welche der Segmente von LD1 und LD2 aufleuchten.

Transistoren

Man unterscheidet grundsätzlich zwischen zwei Arten, nämlich NPN- und PNP-Transistoren. Die Buchstabenfolge PNP und NPN deutet darauf hin, wie im Transistor des Halbleitermaterial angeordnet ist.

Halbleiter der Type n — besitzt einen Überschuß an Elektronen und hat eine negative Ladung

Halbleiter der Type p — hat ein Defizit an Elektronen (oder Überschuß an "Löchern") und eine positive Ladung.

Bild 5-16 zeigt die Anordnung der p- und n-Materialien für PNP- und NPN-Transistoren. Schematisch werden die beiden Transistorarten durch die in Bild 5-17 gezeigten Symbole dargestellt. Die Anschlüsse zu jedem Abschnitt des Halbleitermaterials werden als *Emitter*, *Basis* und *Kollektor* bezeichnet. Die Pfeilrichtung auf dem Emitter gibt an, um welche Transistorart es sich handelt.



Bild 5-16. Halbleiter-Materialverbindungen der Typen N und P.

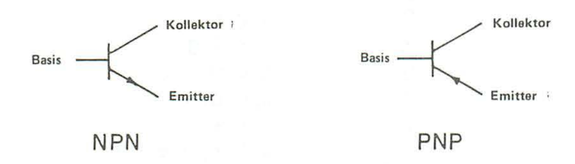


Bild 5-17. Transistor-Symbole.

Transistoren kann man u.a. als elektrisch gesteuerte Schalter betrachten; man schaltet sie ein oder aus, indem der Stromfluß zur Basis verändert wird. Um einen Transistor einzuschalten bzw. in den leitenden Zustand zu versetzen, müssen an den Anschlüssen die richtigen Potentialverhältnisse vorhanden sein. Dieses Verhältnis ist für NPN- und PNP-Transistoren unterschiedlich.

NPN PNP

Bild 5-18. Leitende NPN- und PNP-Transistoren.

NPN PNP

Bild 5-19. Bei den eingezeichneten Potentialverhältnissen sperren die Transistoren.

Bild 5-18 zeigt für den leitenden Zustand beider Transistorarten die richtigen Potentialverhältnisse. Die Pfeilspitze am Emitter deutet auf ein ähnliches Verhalten wie bei einer Diode hin: die Emitter-Anode muß im Verhältnis zur Emitter-Kathode positiv sein, damit Strom fließen kann.
Um einen Transistor zu sperren, braucht man lediglich den Strom zur Basis zu unterbrechen bzw. für Potentialverhältnisse wie in Bild 5-19 zu sorgen.

PNP-TRANSISTOREN IN Q1

Setzt man die grunsätzlichen Transistor-Erläuterungen in die Praxis um, ist die Funktion der Transistor-Arrays in Bild 5-6 leicht einzusehen. Alle Emitter der PNP-Transistoren im IC Q1 (BGY16) sind an +5 V ange-

schlossen. Daher sind sie alle richtig vorgespannt, wenn die Leitungen PB0 . . . PB7 logisch 1 werden. Leitet ein Transistor, ist der Spannungsabfall zwischen Emitter und Kollektor vernachlässigbar klein (0,1 V bis 0,2 V). Daher ist das Potential am Emitter und Kollektor annäherend gleich. Im Sperrzustand ist der Spannungsabfall zwischen Emitter und Kollektor relativ hoch. Vereinfacht läßt sich das Verhalten der PNP-Transistoren in der Nanocomputer®-I/O-Schaltung in Bild 5-6 wie folgt darstellen:

Emitter	Basis	Kollektor
1	0	1
1	1	0

NPN-TRANSISTOREN IN Q2 UND Q3

Die Emitter der 10 NPN-Transistoren sind mit Masse verbunden, daher richtig vorgespannt, wenn die Basis eines jeden einzelnen Transistors logisch 1 wird. In diesem Fall leiten die entsprechenden Transistoren.
Ähnlich wie bei den PNP-Transistoren haben der Kollektor und der Emitter etwa gleiches Potential, wenn der Transistor leitet. Ein Spannungsabfall ist jedoch vorhanden, wenn der Transistor gesperrt ist. Nachstehend eine vereinfachte Wahrheitstabelle für einen NPN-Transistor mit an Masse liegendem Emitter:

Emitter	Basis	Kollektor
0	0	1
0	1	0

Um die logische Operation der Nanocomputer®-I/O-Schaltungen in Bild 5-6 weiter zu definieren, kann man die Transistoren der ICs Q1, Q2 und Q3 als Inverter betrachten. Dabei ist in allen Fällen der Basis-Anschluß der Eingang und der Kollektor-Anschluß der Ausgang.
Selbstverständlich ist mit diesen kurzen Erläuterungen der Transistor nicht erschöpfend erklärt. Eine weitergehende Erklärung ist jedoch für das Verständnis des Nanocomputers® nicht zweckmäßig. Für vollständigere Erklärungen stehen einschlägige Grundsatzbücher zur Verfügung.

4-Bit-COSMOS-Register/4-zu-16-Dekoder HCF4514B (mit logisch 1 am aktivierten Ausgang)

Das IC 4541B ist sowohl ein 4-Bit-Register als auch ein 4-zu-16-Dekoder. Die Pin-Belegung und die Wahrheitstabelle für den Dekoderteil zeigt Tabelle 5-4.
Die vier Registereingänge des ICs sind mit den Leitungen PB1, PB2, PB3 und PB4 verbunden. Das sind vier der acht Leitungen des B-Gatters von PIO 1. Vereinfacht kann man sich Gatter B von PIO 1 als 8-Bit-Auffang-Register vorstellen. Den Eingang bilden die Datenleitungen D0 . . . D7 von der CPU; die Ausgangsleitungen sind PB0 . . . PB7 vom B-Gatter des PIO 1.
PB0 ist an Pin 1 des Register/Dekoders angeschlossen und stellt das Abtastsignal für das 4-Bit-Register dar. Jede negative Signalflanke auf der BUS-Leitung PB0 überträgt die Daten der BUS-Leitungen PB1 . . . PB4 ins Register von IC Q4. Der noch im IC enthaltene Dekoder bestimmt

aufgrund der 4-Bit-Information, welcher Ausgang aktiviert (logisch 1) wird. Jeder der Dekoderausgänge S0 . . . S9 steuert die Basis eines Darlingtontransistors, so daß eine Selektor-LED und die 7-Segmentanzeigen aufleuchten. Die Signale der BUS-Leitungen bestimmen also, welche Ziffer auf dem Display erscheint. Das Register im IC HCF4514B hält die Daten der Leitungen PB1 . . . PB4 solange fest, bis der nächste Taktimpuls PB0 neue Daten ins Register lädt. Das heißt, erst das Abtasten neuer Auswahldaten ändert den bzw. die aktivierten Ausgänge.
Pin 23 des ICs Q4 ist ein Sperreingang. Ist das Eingangssignal logisch 1, sind alle Dekoderausgänge auf logisch 0. Sowohl Pin 1 als auch Pin 23 sind mit der PB0-Leitung verbunden. Das Integrierglied R16/C2 verzögert an Pin 23 das PB0-Signal.
Das Zeitdiagramm in Bild 5-20 zeigt den Einfluß von R16 und C2 auf das Taktsignal an Pin 23 im Gegensatz zum Pin 1. Das Signal an Pin 1 muß für eine relativ lange Zeit logisch 1 bleiben, damit das Signal an Pin 23 ebenfalls logisch 1 wird. Diese Verzögerung gestattet auch kurzen PB0-Impulsen die Übernahme neuer 4-Bit-Daten in das Register, ohne den Dekoderausgang zu blockieren. Dadurch erscheint die Anzeige dem menschlichen Auge ohne störendes Flackern. Auf der einen Seite kann man mit einem längeren PB0-Impuls die Dekoderausgänge für eine bestimmte Zeit blokkieren.

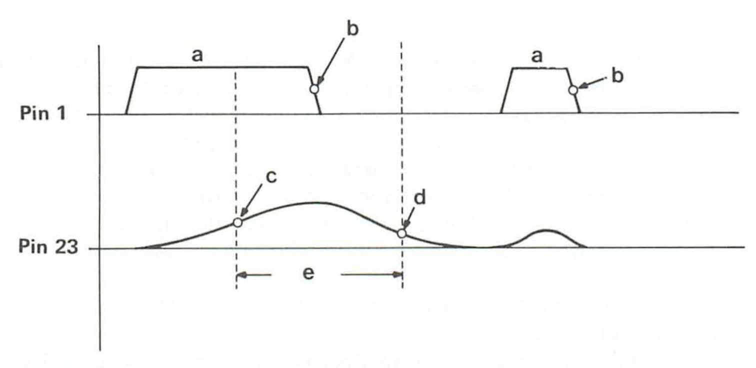


Bild 5-20. Wirkung von R16 und C2 auf das Abtastsignal der Leitung PB0.
a) Freigabe der Dateneingabe, b) Eingabe neuer Daten, c) Übergang von log. 0 auf log. 1, d) Übergang von log. 1 auf log. 0, e) Dekoderausgänge gesperrt.

Das Z-80-PIO-IC

Bei dem Z-80-PIO-IC handelt es sich um eine komplexe, programmierbare, parallele I/O-Steuerung mit zwei Gattern. An dieser Stelle folgt eine Kurzinformation, die ausreicht, Hard- und Software im Zusammenhang mit der Schaltung in Bild 5-6 zu verstehen.

Tabelle 5-4. Eigenschaften des Register/Dekoders HCF4514B

PRELIMINARY DATA

4-BIT LATCH/4-TO-16 LINE DECODER:

HCC/HCF 4514B OUTPUT "HIGH" ON SELECT
HCC/HCF 4515B OUTPUT "LOW" ON SELECT

- QUIESCENT CURRENT SPECIFIED TO 20V
- MAX. INPUT LEAKAGE CURRENT 1 μA @ 18V (FULL PACKAGE - TEMP. RANGE)
- STROBED INPUT LATCH
- INHIBIT CONTROL

The **HCC 4514B/HCC 4515B** (extended temperature range) and the **HCF 4514B/HCF 4515B** (intermediate temperature range) are monolithic integrated circuits available in 24-lead dual in-line plastic and ceramic slam package. The **HCC/HCF 4514B/4515B** consisting of a 4-bit strobed latch and a 4 to 16 line decoder. The latches hold the last input data presented prior to the strobe transition from 1 to 0. Inhibit control allows all outputs to be placed at 0 (**HCC/HCF 4514B**) or 1 (**HCC/HCF 4515B**) regardless of the state of the data or strobe inputs. The decode truth table indicates all combinations of data inputs and appropriate selected outputs.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{DD}*	Supply voltage	-0.5 to 20	V
V_I	Input voltage	-0.5 to V_{DD} +0.5	V
I_I	DC input current (any one input)	± 10	mA
P_{tot}	Total power dissipation (per package)	200	mW
	Dissipation per output transistor for T_{op} = full package-temperature range	100	mW
T_{op}	Operating temperature: for **HCC** types	-55 to 125	°C
	for **HCF** types	-40 to 85	°C
T_{stg}	Storage temperature	-65 to 150	°C

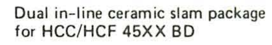
* All voltage values are referred to V_{SS} pin voltage

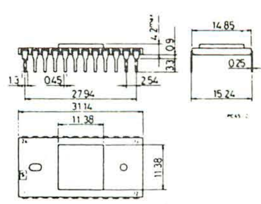
ORDERING NUMBERS:

HCC 45XX BD for dual in-line ceramic slam package
HCF 45XX BD for dual in-line ceramic slam package
HCF 45XX BE for dual in-line plastic package

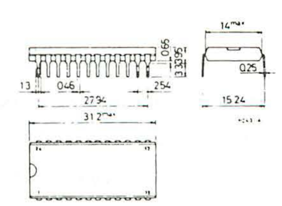
MECHANICAL DATA

dimensions in mm

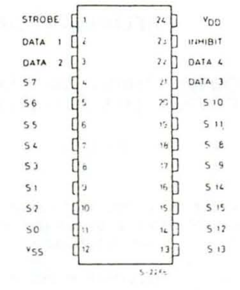

Dual in-line ceramic slam package
for HCC/HCF 45XX BD



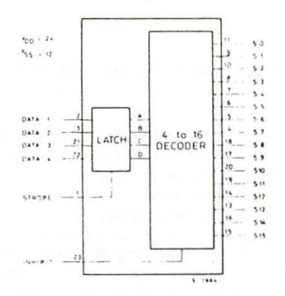
Dual in-line plastic package
for HCF 45XX BE



CONNECTION DIAGRAM



FUNCTIONAL DIAGRAM



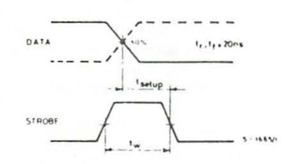
DECODER TRUTH TABLE*

INHIBIT	DATA INPUTS				SELECTED OUTPUT HCC/HCF 4514B= Logic 1 (High) HCC/HCF 4515B= Logic 0 (Low)
	D	C	B	A	
0	0	0	0	0	S0
0	0	0	0	1	S1
0	0	0	1	0	S2
0	0	0	1	1	S3
0	0	1	0	0	S4
0	0	1	0	1	S5
0	0	1	1	0	S6
0	0	1	1	1	S7
0	1	0	0	0	S8
0	1	0	0	1	S9
0	1	0	1	0	S10
0	1	0	1	1	S11
0	1	1	0	0	S12
0	1	1	0	1	S13
0	1	1	1	0	S14
0	1	1	1	1	S15
1	X	X	X	X	All Outputs= 0, HCC/HCF 4514B All Outputs= 1, HCC/HCF 4515B

X = Don't Care *Strobe = 1
1 = high
0 = low

WAVEFORMS

Setup time and strobe pulse width



RECOMMENDED OPERATING CONDITIONS

V_{DD}	Supply voltage	3 to 18	V
V_I	Input voltage	0 to V_{DD}	V
T_{op}	Operating temperature: for **HCC** types	-55 to 125	°C
	for **HCF** types	-40 to 85	°C

Tabelle 5-4. Fortsetzung

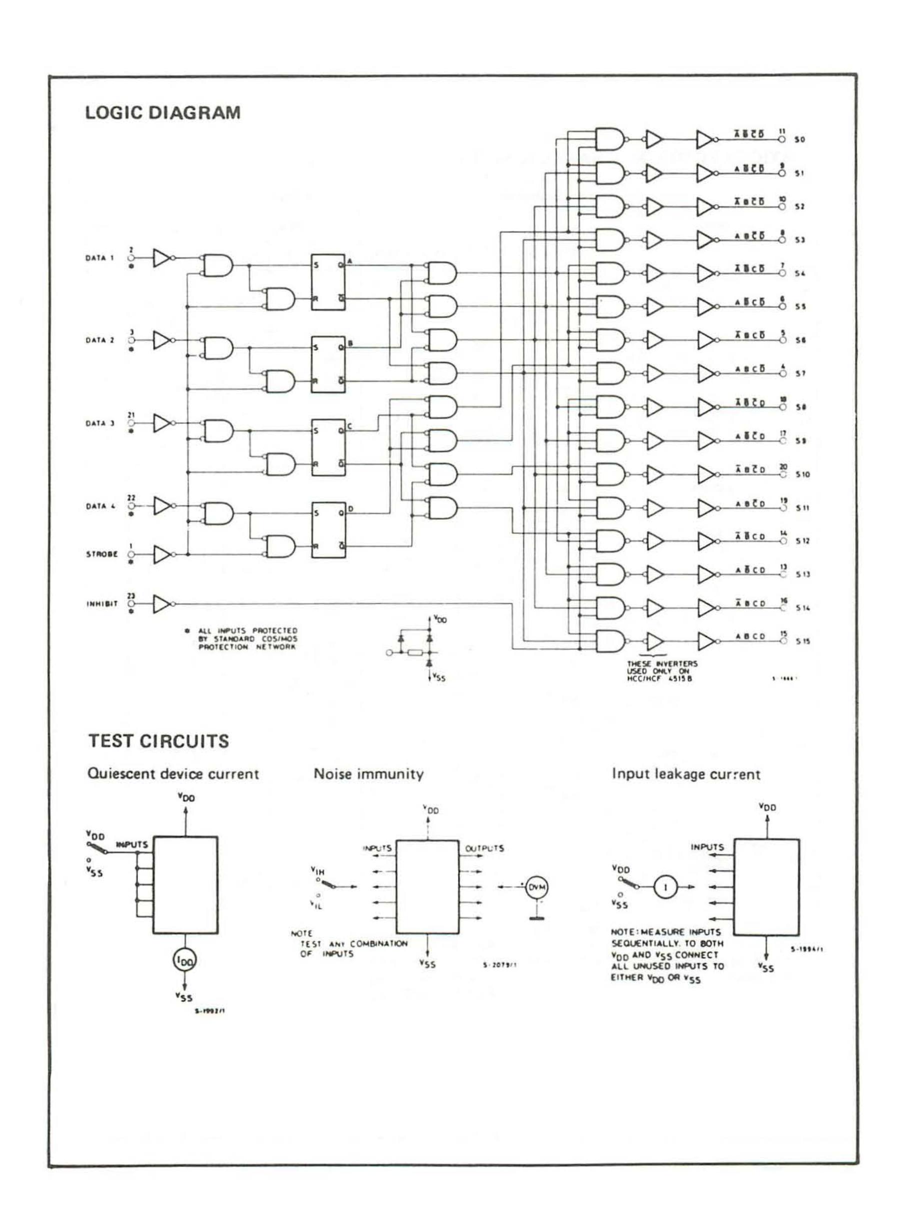


Tabelle 5-4. Fortsetzung

STATIC ELECTRICAL CHARACTERISTICS (over recommended operating conditions)

Parameter			Test conditions V_I (V)	V_O (V)	$\lvert I_O \rvert$ (μA)	V_{DD} (V)	Values T_{Low}* Min.	T_{Low}* Max.	25°C Min.	25°C Typ.	25°C Max.	T_{High}* Min.	T_{High}* Max.	Unit
I_L	Quiescent supply current		0/ 5			5		1		0.02	5		150	μA
			0/10			10		2		0.02	10		300	
			0/15			15		4		0.02	20		600	
			0/20			20		20		0.04	100		3000	
V_{OH}	Output high voltage		0/ 5		< 1	5	4.95		4.95			4.95		V
			0/10		< 1	10	9.95		9.95			9.95		
			0/15		< 1	15	14.95		14.95			14.95		
V_{OL}	Output low voltage		5/0		< 1	5		0.05			0.05		0.05	V
			10/0		< 1	10		0.05			0.05		0.05	
			15/0		< 1	15		0.05			0.05		0.05	
V_{IH}	Input high voltage			0.5/4.5	< 1	5	3.5		3.5			3.5		V
				1/9	< 1	10	7		7			7		
				1.5/13.5	< 1	15	11		11			11		
V_{IL}	Input low voltage			4.5/0.5	< 1	5		1.5			1.5		1.5	V
				9/1	< 1	10		3			3		3	
				13.5/1.5	< 1	15		4			4		4	
I_{OH}	Output drive current	HCC types	0/ 5	2.5		5	-2		-1.6	-3.2		-1.15		mA
			0/ 5	4.6		5	-0.64		-0.51	-1		-0.36		
			0/10	9.5		10	-1.6		-1.3	-2.6		-0.9		
			0/15	13.5		15	-4.2		-3.4	-6.8		-2.4		
		HCF types	0/ 5	2.5		5	-1.8		-1.6	-3.2		-1.3		
			0/ 5	4.6		5	-0.61		-0.51	-1		-0.42		
			0/10	9.5		10	-1.5		-1.3	-2.6		-1.1		
			0/15	13.5		15	-4		-3.4	-6.8		-2.8		
I_{OL}	Output sink current	HCC types	0/ 5	0.4		5	0.64		0.51	1		0.36		mA
			0/10	0.5		10	1.6		1.3	2.6		0.9		
			0/15	1.5		15	4.2		3.4	6.8		2.4		
		HCF types	0/ 5	0.4		5	0.61		0.51	1		0.42		
			0/10	0.5		10	1.5		1.3	2.6		1.1		
			0/15	1.5		15	4		3.4	6.8		2.8		
I_{IH}, I_{IL}**	Input leakage current		0/18			18		±0.1		$\pm 10^{-5}$	±0.1		± 1	μA
C_I**	Input capacitance									5	7.5			pF

* T_{Low} = - 55°C for HCC device; - 40°C for HCF device.
* T_{High} = +125°C for HCC device; + 85°C for HCF device.
The Noise Margin for both "1" and "0" level is: 1V min. with V_{DD} = 5V; 2V min. with V_{DD} = 10V; 2.5V min. with V_{DD} = 15V
** Any input

Auf der Platine des Nanocomputers® befinden sich zwei PIO-ICs, die mit PIO1 und PIO2 bezeichnet sind. PIO1 führt die I/O-Funktion der Tastenfeld- und Anzeigeeinheit durch, während PIO2 dem Anwender zur freien Verfügung steht. Der folgende Abschnitt beschäftigt sich mit dem PIO1. Kapitel 7 beschreibt die Benutzung von PIO2; zur der theoretischen Beschreibung gehören ausführliche Versuche.
Die beiden Gatter im PIO-IC werden im allgemeinen mit Gatter A und Gatter B bezeichnet. Bei beiden Gattern kann man durch entsprechende Programmierung zwischen vier verschiedene Operationsarten auswählen. Die Programmierung schließt die Ausgabe definierter Byteworte mit ein. Sie bestimmen über je ein separates Steuergatter die Operationsart für Gatter A und Gatter B. Zum PIO1 gehören also vier I/O-Gatter.

Gatter	Hex-Gerätecode
Gatter A, Daten	04
Gatter B, Daten	05
Gatter A, Steuerung	06
Gatter B, Steuerung	07

Die Gerätecodes 08, 09, 0A und 0B haben eine ähnliche Beziehung zu PIO2.
Die vier Operationsarten sind folgende:

Methode 0: Byte-Ausgabe: für Parallelausgabe durch Unterbrechung
Methode 1: Byte-Eingabe: für Paralleleingabe durch Unterbrechung
Methode 2: bidirektionales Byte: nur für bidirektionale I/O an Gatter A
Methode 3: Steuermethode: Sowohl für Ein- als auch für Ausgabe. Jedes der acht Gatter-Bits ist als Ein- oder Ausgabe-Bit bestimmt. Der PIO-IC arbeitet im Grunde wie ein Auffang-Register für Ausgabe-Bits und wie ein Puffer und Auffang-Register für Eingabe-Bits. (Die Steuermethode bietet noch weitere wirksame Möglichkeiten, auf die später eingegangen wird.)

Über eine Routine im Operationssystem des Nanocomputers® wird PIO1 gestartet oder programmiert, um sowohl Gatter A als auch B nach der Steuermethode (Methode 3) zu betätigen. Dabei werden alle Bits des Gatters B als Ausgangs-Bits und alle Bits des Gatters A wie folgt bezeichnet: I000IIII. Somit sind die Bits D7, D3, D2, D1 und D0 Eingabe-Bits und die Bits D6, D5, und D4 Ausgabe-Bits. Die Ausgabe-Bits von Gatter A sind also wie folgt definiert:

D7	D6	D5	D4	D3	D2	D1	D0
I	0	0	0	I	I	I	I

Dies bedeutet im wesentlichen, durch die Bauweise von PIO1 legt der Befehl OUT (05H), A alle sieben Bits des Z-80-Daten-BUS auf den Daten-BUS PB0 . . . PB7 des Gatters B, und der Befehl OUT (04H), A legt D4, D5 und D6 auf den Daten-BUS des Gatters A: PA4, PA5 und PA6. Ein IN-Befehl ergibt für Gatter B keinen Sinn, da alle seine Datenleitungen für Ausgabe vorgesehen sind. Aber ein Befehl IN A, (04H) liest alle Daten auf den Leitungen PA0, PA1, PA2, PA3 und PA7 in den Akkumulator ein.

Software

Betrachten Sie im Schaltbild 5-6 die Verbindungen zum Daten-BUS PB0 . . . PB7 des Gatters B.

Die Leitungen zum Gatter A werden bei der Beschreibung der Tastenfeld-Eingabe des Nanocomputers® näher betrachtet. Zunächst ist folgender Funktionsablauf interessant:

1. Setzen Sie die Bits PB1 . . . PB4 auf 0000 über Befehle wie

```
XOR A
OUT (05H), A
```

2. Schalten Sie das PB0-Bit durch eine Befehlsfolge wie

```
INC A
OUT (05H), A
DEC A
OUT (05H), A
```

   Während die Bits D1 . . . D4 (tatsächlich D1 . . . D7) konstant bleiben, passiert auf der Leitung D0 etwas. Der auf der Leitung anstehende Impuls (Übergang von low nach high und wieder nach low) überträgt sich auf die Leitung PB0, wodurch das Register im IC HCF4514B die Signalpegel der Leitungen PB1 . . . PB4 übernimmt. Verantwortlich dafür ist die negative Impulsflanke auf PB0. Durch die Übernahme von 0000 in den Dekoder wird der Ausgang S0 (Pin 11) aktiviert. Mit dem Ausgang verbunden ist die Basis eines Darlington-NPN-Transistors im IC Q2. Der Transistor schaltet durch und wählt für die Anzeige die LED-Reihe L1 . . . L7 aus.

3. Soll nun aus der LED-Reihe L1 . . . L7 die Selektor-LED MEM aufleuchten, muß der entsprechende Transistor im IC Q1 leitend werden. Das heißt, der Kollektor muß den Zustand logisch 1 annehmen. Dazu ist an der Basis des PNP-Transistors eine "0" erforderlich. Alle anderen Transistoren im IC Q1 müssen sperren. Die Bitfolge auf den Leitungen PB1 . . . PB7 muß also (das entspricht der Hex-Ziffer DE) lauten 1101111. Denn:
   PB7: logisch 1, damit L1, die BRK-LED, nicht aufleuchtet
   PB6: logisch 1, damit L2, die I/O-LED, nicht aufleuchtet
   PB5: logisch 0, damit L3, die MEM-LED, aufleuchtet
   PB4: logisch 1, damit L4, die PC-LED, nicht aufleuchtet
   PB3: logisch 1, damit L5, die SP-LED, nicht aufleuchtet
   PB2: logisch 1, damit L6, die ERR-LED, nicht aufleuchtet
   PB1: logisch 1, damit L7, die ARS-LED, nicht aufleuchtet

Der Abtastimpuls auf der PB0-Leitung darf nur von relativ kurzer Dauer sein, da ansonsten der Sperreingang von IC Q4 ebenfalls logisch 1 wird und die Ausgänge S0 . . . S9 blockiert.

Die folgenden Befehle legen die gewünschten Werte auf den Ausgangs-BUS von Gatter B:

```
LD A, DEH
OUT (05H), A
```

Im Register des ICs HCF4514B bleibt die gewählte Information für die Selektor-LED erhalten (im Beispiel die LED-Reihe L1 . . . L7). Hingegen wird der Daten-BUS von Gatter B (vorher zur Anzeige der Selektor-LED MEM benutzt) nun eine völlig andere Aufgabe übernehmen. Er bestimmt nämlich die Anzeige auf den 7-Segment-Displays.

Der gerade beschriebene Funktionsablauf entspricht der Ausgangs-Treiberroutine des Nanocomputers®, um die Selektor-LEDs und die Hex-Ziffern auf dem Display aufleuchten zu lassen. Was ist, wenn z.B. die beiden LEDs I/O und MEM gleichzeitig aufleuchten sollen? Der Funktionsablauf und das Verfahren ist in beiden Fällen gleich. Es ändert sich lediglich der Befehl LD A, DEH in LD A, 9EH. Dadurch wird sowohl die I/O- als auch die MEM-LED richtig vorgespannt. Eine andere Problemstellung ist folgende: Es soll z.B. die Selektor-LED MEM und eine bzw. mehrere Siebensegment-Anzeigen oder eine LED der Reihe L8 . . . L14 aufleuchten. Durch den Austausch des letzten Ausgabe-Bytes läßt sich diese Aufgabe nicht lösen, weil zwei verschiedene Anzeigen scheinbar gleichzeitig angewählt werden müssen. Außerdem sind zwei verschiedene Daten-Bytes erforderlich; einmal wegen der Selektor-LED MEM und zum zweiten wegen der anderen Anzeige. Wie läßt sich dies nun bewerkstelligen?
Im Prinzip überhaupt nicht. Der Dekoder im IC Q4 in Bild 5-6 kann im selben Augenblick *nicht gleichzeitig* zwei Ausgänge aktivieren. Die Schaltung ist jedoch in der Lage, zwei Ausgänge im sehr schnellen Wechsel zu aktivieren, ohne daß das relativ träge menschliche Auge diesen Wechsel bemerkt. Die CPU kann zusammen mit der Schaltung Selektor-LEDs und Ziffern auf dem Display im sehr schnellen Wechsel aufleuchten lassen. Dadurch entsteht der Eindruck, daß die Anzeigen gleichzeitig aktiv sind. Selbst wenn eine Anzeige alle acht 7-Segment-Displays und beide LED-Reihen umfaßt, ist der Wechsel so schnell, daß vom menschlichen Auge nichts wahrgenommen wird. Dieses Auffrischen im Multiplexverfahren spart Bauelemente, Platz und somit auch Kosten. Das Auffrischen ist bei vielen Sichtgeräten eine bekannte Tatsache, so z.B. beim TV-Gerät, bei der Kathodenstrahlröhre und bei den Datensichtgeräten. Wird ein Buchstabe oder eine Zahl in der Sekunde ca. 60 mal aufgefrischt, nimmt das menschliche Auge nicht einmal ein Flackern wahr. Die Anzahl der Wiederholungen nennt man *Auffrischrate.*
Bei dem folgenden Programm handelt es sich um einen einfachen Anzeigentreiber. Das Programm demonstriert, wie man mehr als nur eine Anzeige gleichzeitig aufleuchten läßt. Der Akkumulator ist der Anzeigewähler, während das B-Register das anzuzeigende Daten-Byte enthält. Der Akkumulator-Inhalt wird zweimal bei jeder Schleife (unmittelbar bevor eine Verzögerungsschleife aufgerufen wird) erhöht. Die eigentliche Anzeigeauswahl übernehmen die Bits D1 . . . D7 (oder PB1 . . . PB7), so daß anstatt einer zwei Erhöhungsanweisungen notwendig sind. Die Erhöhung von D0 . . . D7 um 2 entspricht daher der Erhöhung von D1 . . . D7 um 1. Das B-Register verändert sich niemals! So stellt es je Anzeige dasselbe Daten-Byte dar. Natürlich ist die Anzeige der beiden LED-Reihen optisch anders als die der 7-Segment-Displays. Wie aus dem Schaltbild 5-6 hervorgeht, können die LEDs L1 und L8 sowie die Segmente a von jedem Display gleichzeitig aufleuchten.
Um alle 10 Anzeigen (8 Displays und zwei LED-Reihen) nacheinander zu aktivieren, zählt das D-Register von 10 bis 0 rückwärts. Ist der Inhalt von Register D Null, folgt ein Sprung zurück zum Anfang der Anzeige-Treiberroutine (Anweisung BEGIN). Dadurch läuft das Programm erneut ab, so daß sich der Anzeige-Zyjklus wiederholt. Zum Abschluß dieses Kapitels werden Sie mit einigen Varianten zu diesem Programm einen Versuch durchführen.

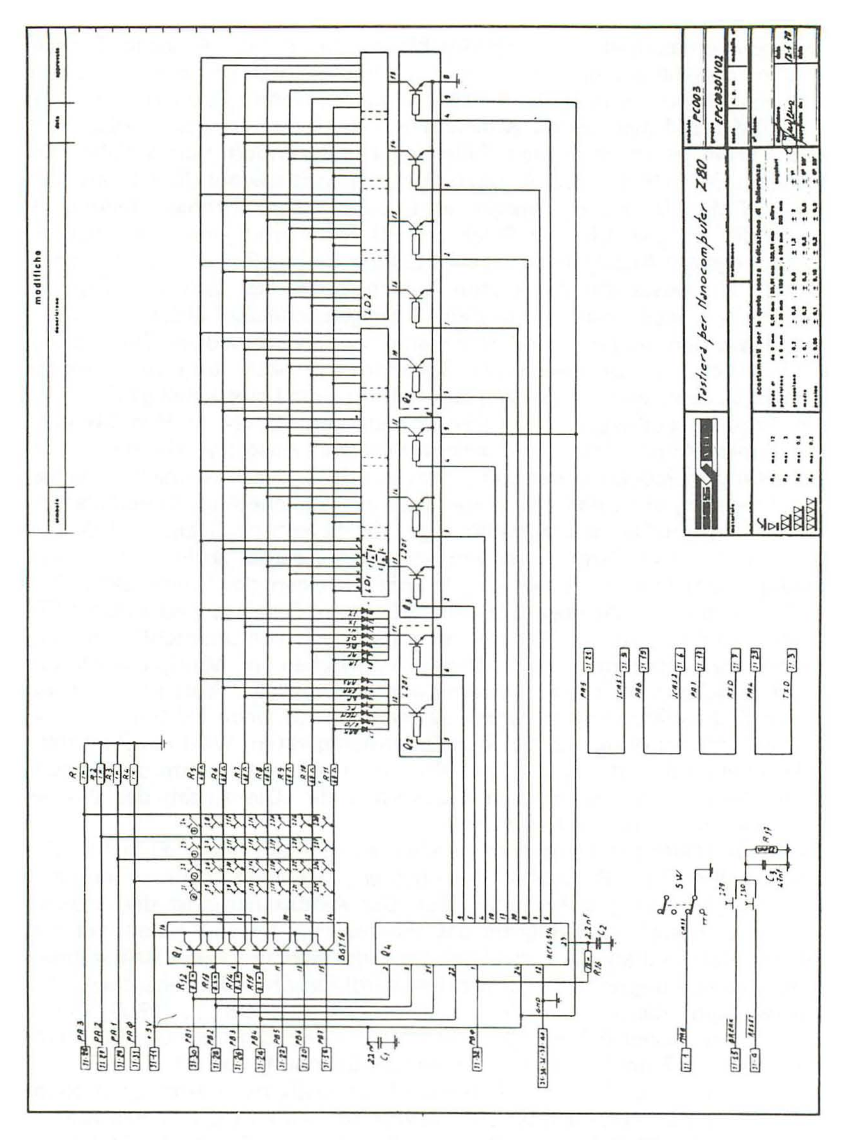


Bild 5-21. Tastenfeld- und Anzeige-Hardware des Nanocomputers®.

Ein späterer Abschnitt über das Software-Operationssystem des Nanocomputers® befaßt sich auch mit dem Anzeigetreiber. Er wird vom Operationssystem benutzt, um die der gedrückten Taste entsprechenden Anzeige auszugeben. Doch ist zunächst die Eingangsschaltung aus Bild 5-6 und die dazugehörige Software von Interesse.

```
BEGIN:      LD D,0AH          ; Zähler D auf 0AH setzen
            XOR A             ; Anzeigewähler aktivieren
            LD BC,7E05H       ; das Daten-Byte 7E und das
                              ; Ausgangs-Gatter 05 setzen
                              ;
OUTPUT:     OUT (C), A        ; Anzeigewähler ausgeben
            INC A             ; durch Schalten von Bit D0
            OUT (C), A        ; Byte zum Dekoder/Auffang-Reg.
            DEC A
            OUT (C), A
            OUT (C), B        ; Daten-Byte ausgeben
            INC A             ; Register A auf neusten Stand
            INC A             ; bringen, um nächste Anzeige
                              ; festzulegen
            CALL DELAY        ; eine Weile verzögern
            DEC D             ; D erniedrigen um zu sehen, ob
                              ; Zyklus beendet ist
            JR NZ,OUTPUT      ; falls D nicht 0, Zyklus fortführen
            JR BEGIN          ; falls D = 0, zurück zur Anweisung
                              ; BEGIN
```

TASTENFELD-EINGABESYSTEM DES NANOCOMPUTERS®

Hardware

Zum besseren Verständnis ist das Schaltbild der Tastenfeld- und Anzeige-Hardware des Nanocomputers® in Bild 5-21 nochmals gezeigt. Folgende Bauelemente sind an den Tastenfeld-Eingabeoperationen beteiligt:

Q1: BGY16 – IC mit sieben getrennten PNP-Transistoren.
I1 . . . I28: 28 Tasten auf dem Tastenfeld des Nanocomputers®.

Damit ist klar, daß an der Eingabe weniger Bauelemente als bei der Anzeige beteiligt sind. Das IC Q1 (BGY16) ist bereits bei der Anzeige-Funktion beschrieben. Was also noch zu erläutern ist, sind die 28 Tasten des Tastenfeldes.

Das 28-Tastenfeld I1 . . . I28

Das Tastenfeld ist in sieben Reihen und vier Spalten aufgeteilt. Für die Entprellung, Erkennung und Interpretation der gedrückten Tasten sind im Operationssystem des Nanocomputers® mehrere Software-Routinen zuständig. Der Entprellvorgang ist im Prinzip nur eine Verzögerung der Software-Routine, die einmal eine gedrückte Taste erkennt und zum anderen der Tastenfunktion Zeit gibt, sich im gedrückten oder ungedrückten Zustand zu stabilisieren.

Die Tasten sind nicht hardwaremäßig entprellt. Daher muß der Übergang vom ungedrückten in den gedrückten Zustand oder umgekehrt immer verzögert erfolgen. Ansonsten kann der Nanocomputer® durch eventuelles Kontaktprellen eine Tasteneingabe falsch intepretieren. Das Einfügen einer kurzen Verzögerungsschleife in die Software definiert den Tastenzustand "Taste gedrückt" eindeutig und schaltet damit alle Probleme durch nicht zulässiges Kontaktprellen aus. Auf diese Weise liest die Software alle Tasteninformationen im gefestigten Zustand, was die Voraussetzung für verläßliche Tastenfeld-Eingabedaten sind.

Software

Das Betriebssystem des Nanocomputers® erkennt gedrückte Tasten in zwei Stufen. In der ersten Stufe wendet es eine Routine an, die CHECKB genannt wird, um zu sehen, ob *irgendeine* Taste gedrückt ist. Für die zweite Stufe ist die KBSCAN-Routine zuständig, die folgendes bestimmt:

a) Ist nur *eine* Taste gedrückt? Falls mehr als eine Taste gedrückt ist, werden alle gedrückten Tasten ignoriert.

b) Falls nur eine Taste gedrückt ist, welche ist es? Die Tasten sind numeriert von 1 . . . 28, daher wird eine Zahl zwischen 1 und 28 als Antwort zurückgegeben.

Zur Erkennung von gedrückten Tasten wenden beide Routinen dieselbe Grundtechnik an. Eine oder mehrere Bits von PB1 . . . PB7 nehmen den Zustand logisch 0 an, wodurch die Kollektoren der entsprechenden PNP-Transistoren in Q1 ebenfalls den Zustand logisch 1 annehmen. Aus der im Schaltbild 5-21 gezeichneten Darstellungsart ist leicht zu erkennen, wie eine gedrückte Taste eine Ausgabeleitung des Gatters B mit einer Eingabeleitung des Gatters A verbindet. Eine logische 1 auf je einer der PA-Leitungen PA0 . . . PA3 bedeutet, daß eine Taste gedrückt ist. Die Routine CHECKB arbeitet nach dem "Schrotflintenprinzip". Jede der Leitungen PB1 . . . PB7 wird mit einem Null-Bit belegt, so daß die Tastenkontakte der horizontalen Leitungen eine logische 1 führen. Durch die Ausführung eines Eingabebefehls IN A, (04H) von Gatter A und auf der Suche nach einer logischen 1 auf einer der vier Eingabeleitungen ist CHECKB in der Lage, eine gedrückte Taste zu erkennen. Wie bereits erwähnt, arbeitet Gatter A von PIO1 nach der Steuermethode (Methode 3) und operiert mit den I/O-Bits 10001111. Die Eingabe-Bits PA1 . . . PA3 sind im PIO1 gespeichert; die entsprechenden Gerätecodes gelten für die Daten von Gatter A (04) und für die Steuerung Gatter A (06).

Diese Informationen reichen jedoch noch nicht aus, um die gedrückte Taste genau zu erkennen. Dazu ist noch die Routine KBSCAN erforderlich. Wie auch immer, die Routine CHECKB ist schnell, unkompliziert und somit sehr zweckdienlich. Die Routine läuft immer wieder ab, um einen eventuellen Tastendruck zu registrieren. Im Gegensatz dazu ist der Prozentsatz positiver Prüfungen relativ gering. Der folgende Abschnitt beschreibt die CHECKB-Routine so, wie sie im Operationssystem des Nanocomputers® erscheint.

SYSTEM-SOFTWARE-DOKUMENTATION TASTENFELD-EINGABEROUTINEN: CHECKB, IO, UND KBSCAN

SUBROUTINE CHECKB

Sie hat die Aufgabe festzustellen, ob eine Taste gedrückt ist. Das Zero-Flag wird zurückgesetzt, wenn man eine oder mehrere Tasten gedrückt hat, ansonsten ist es gesetzt.

Speicherstelle

CHECKB (absolute Speicherstelle siehe Anhang A, Tabelle A-1).

Eingabedaten

Keine.

Ausgabedaten

Bit 0: geht auf logisch 1 (wird gesetzt), wenn man eine Taste der Spalte 1 betätigt hat, ansonsten zurückgesetzt.

Bit 1: geht auf logisch 1, wenn man eine Taste der Spalte 2 betätigt hat, ansonsten zurückgesetzt.

Bit 2: geht auf logisch 1, wenn man eine Taste der Spalte 3 betätigt hat, ansonsten zurückgesetzt.

Bit 3: geht auf logisch 1, wenn man eine Taste der Spalte 4 betätigt hat, ansonsten zurückgesetzt.

Das Zero-Flag wird logisch 0, wenn man eine oder mehrere Tasten gedrückt hat, ansonsten ist es gesetzt.

Tabelle 5-5. Anordnung des Nanocomputer®-Tastenfeldes (Row = Reihe, Col. = Spalte)

	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	0	1	2	3
Row 2	4	5	6	7
Row 3	8	9	A	B
Row 4	C	D	E	F
Row 5	→	←	ST	LA
Row 6	2ND	SS	INC	LD
Row 7	ARS	GO	BRK	DP

Benutzte Register

Es sind hier nur die Register aufgeführt, deren Inhalte sich während der laufenden Subroutine verändern. Das heißt: Nach Ablauf der Subroutine sind die vom Anwender in den erwähnten Registern abgelegten Daten verändert. Das gilt auch bei den Beschreibungen der folgenden Subroutinen.

Beschreibung

Als erster Schritt werden alle horizontalen Leitungen (siehe Bild 5-21) in der Tastenfeld-Matrix eingeschaltet und das Dekoder/Register zurückgesetzt. Dafür sorgt die Hex-Ziffer bei Gatter B (05) von PIO1. Dadurch steht an jedem Kollektor der Transistoren im IC Q1 (BGY16) eine logische 1 zur Verfügung. Dieser high-Signalpegel gelangt über jede gedrückte Taste zur Eingangsleitung bei Gatter 04 (Daten-Gatter A in PIO1), entsprechend der Spalte der gedrückten Taste.

Nach einer Verzögerung von 17 Mikrosekunden wird der Inhalt von Gatter 04 in den Akkumulator eingelesen und über eine AND-Funktion

mit der Hex Ziffer OF verknüpft. Das ist notwendig, um das Zero-Flag richtig zu setzen. Die Steuerung geht anschließend an die abrufende Routine zurück.

SUBROUTINE IO

Sie gibt ein Byte an die Transistoren im IC BGY16 aus und prüft, ob eine der Tasten gedrückt ist.

Speicherstelle

IO (ist in der Subroutine CHECKB enthalten; absolute Adresse siehe Anhang).

Eingabedaten

Der Akkumulator enthält das an die Transistoren in BGY16 anzugebende Byte.

Ausgabedaten

Die vier niederwertigsten Bits in Akkumulator werden wie folgt gesetzt:
Bit 0: Setzen, falls die gedrückte Taste aus Spalte 1 mit einem leitenden Transistor verbunden ist. Sonst zurücksetzen.
Bit 1: Setzen, falls die gedrückte Taste aus Spalte 2 mit einem leitenden Transistor verbunden ist. Sonst zurücksetzen.
Bild 2: Setzen, falls die gedrückte Taste aus Spalte 3 mit einem leitenden Transistor verbunden ist. Sonst zurücksetzen.
Bit 3: Setzen, falls die gedrückte Taste aus Spalte 4 mit einem leitenden Transistor verbunden ist. Sonst zurücksetzen.
Das Zero-Flag wird zurückgesetzt (logisch 0), wenn eine gedrückte Taste erkannt ist. Andernfalls geht das Flag auf logisch 1.

Benutzte Register

A, F

Beschreibung

Vor Abruf der Subroutine IO erhält der Akkumulator eine Information die festlegt, welche Tastenfeldreihen auf gedrückte Tasten überprüft werden sollen. Eine logische 0 in Bit Dn leitet die Prüfung der Reihe n ein, wobei n = 1 . . . 7. (Bit D0 wird zum Abtasten des Dekoder/Register HCF4514 benutzt und ist daher in dieser Routine unbedeutend!) Die logischen Werte des Akkumulators von Bit D1 . . . D7 für die Prüfung gedrückter Tasten in den Reihen 1 . . . 7 haben folgende Bedeutung:

- Die mit der Basis der Transistoren in Q1 (BGY16) verbundenen Leitungen sind PB1 . . . PB7; auf ihnen stehen die Bits der Ausgangsgatter 05 zur Verfügung (Gatter 05 entspricht dem Daten-BUS-Gatter B in PIO1).
- Der Befehl OUT (05H), A gibt den Akkumulator-Inhalt auf die Leitungen PB1 . . . PB7.
- Ein low-Spannungspegel an der Basis eines der Transistoren im IC Q1 versetzt den entsprechenden Transistor in den leitenden Zustand, d.h. der Kollektor führt logisch 1.

- Eine logische 1 am Kollektor eines Transistors in IC Q1 gelangt über einen geschlossenen Tastenkontakt zu einem der vier Eingabe-Bits des Eingang-Gatters 04 das dem Daten-BUS-Gatter A in PIO1 entspricht). Die gesetzten Bits bei Gatter 04 entsprechen der Spalte einer jeden gedrückten Taste.
- Der Befehl IN A, (04H) liest die vier Spalten-Status-Bits in den Akkumulator ein und untersucht sie auf das Vorhandensein einer logischen 1, was einer gedrückten Taste entspricht.

Die geschilderten Tatsachen machen die Befehle für die Subroutine IO verständlich.

Listing für CHECKB und IO

```
CHECKB: LD A, 01H           ; Register A ist das Ausgabe-Byte; da D0
                            ; high ist, werden die Anzeigen gesperrt.
        EX(SP),HL           ; 17 Mikrosekunden Verzögerung
        EX(SP),HL
IO:     OUT (05H),A         ; PB1 . . . PB7 werden in den Zustand
                            ; logisch 0 versetzt, wodurch alle horizon-
                            ; talen Tastenfeld-Leitungen auf high
                            ; gehen.
        EX(SP),HL           ; 17 Mikrosekunden Verzögerung
        EX(SP),HL
        IN A,(04H)          ; die vier Eingangsleitungen PA0 . . . PA3
        AND 0FH             ; lesen, falls Tasten gedrückt sind, wird
                            ; das Flag Z zurückgesetzt, sonst gesetzt.
        RET                 ; zum Abrufprogramm zurückkehren.
```

Die Verzögerung vor dem Befehl OUT (05H), A dient dem bereits erwähnten Unterdrücken von Kontaktprellen. Nach dem Befehl gibt die Verzögerung den Transistoren im IC Q1 genügend Zeit leitend zu werden.

SUBROUTINE KBSCAN

Die Routine hat die Aufgabe, die Tastatur nach gedrückten Tasten abzusuchen, das Vorhandensein einer einzelnen gedrückten Taste zu erkennen und ihre Identität zu der Abrufroutine zurückzugeben.

Speicherstelle

KBSCAN (absolute Speicherstelle siehe Anhang A, Tabelle A-1).

Eingabedaten

Keine.

Ausgabedaten

Das Carry-Flag wird gesetzt, wenn keine oder mehr als eine Taste gedrückt ist; es geht auf logisch 0, wenn man genau nur eine Taste drückt. Den Wert der gedrückten Taste übernehmen die Register A und C.

Benutzte Register

A, C, F und B.

Beschreibung

Um jede Reihe auf das Vorhandensein einer gedrückten Taste abzusuchen, müssen Register B und C zuerst initialisiert (auf den aktuellen Stand gebracht) werden. Nur dann inkrementiert Register B bei einer gedrückten Taste (z.B. Reihe 1) ordnungsgemäß und Register C kann die Reihe 1 (Bit D0 gesetzt) auswählen. Diesen Test übernimmt die Subroutine IO, wobei der Akkumulator mit dem Komplement des Inhalts von Register C geladen wird. Ist keine Taste gedrückt, rotiert das in Register C gesetzte Bit in das Carry-Flag. Ist das rotierte Bit logisch 1, folgt die Rückkehr ins Abrufprogramm (RET C). Ist eine Taste als gedrückt erkannt, geht das Zero-Flag auf logisch 0, so daß die IC-Routine eine Schleife ausführt. Dabei dient die Information im Akkumulator dazu, Register B so oft zu inkrementieren, bis der Inhalt die Zahl der Spalte wiedergibt, in der sich die gedrückte Taste befindet. Ein weiterer Test lokalisiert mehrere gedrückte Tasten. Dabei wird überprüft, ob ein Akkumulator-Bit 0 . . . 3 gesetzt ist, sobald daß am weitesten rechts stehende gesetzte Bit in das Carry-Flag rotiert ist. Der Test bedient sich einer logischen AND-Operation zwischen dem rotierten Akkuinhalt und dem Byte 0F. Der Test ist negativ, wenn das Ergebnis der AND-Operation genau den Akkumulator-Bits 0 . . . 3 entspricht, d.h. es ist nur eine Taste gedrückt. Ist das Testergebnis positiv, sind mehrere Tasten gedrückt, übernimmt das Abrufprogramm mit gesetztem Carry-Flag die weitere Steuerung.
Die nächste Gruppe Gruppe von Anweisungen bis zum LPKB1-Test hat die Aufgabe festzustellen, ob eine Taste in einer anderen Reihe gedrückt ist. Dazu wird der Inhalt von Register C (anstelle des logischen Komplements von C) an Gatter 04 ausgegeben. Dieses Verfahren prüft alle Reihen auf das Vorhandensein einer gedrückten Taste, sobald die Befehle IN A, (04H) und AND 0FH zur Ausführung kommen. Ausgenommen hiervon ist die Reihe, deren Bit in Register C gesetzt ist. Für die Tasten-Entriegelung und für die Transistor-Schaltzeiten ist eine Verzögerung von 30 Mikrosekunden vorgesehen. Wiederum verursacht ein positives Resultat der AND-Operation eine Rückkehr bei gesetztem Carry-Flag. Falls in den anderen Reihen keine Tasten gedrückt sind, handelt es sich nur um eine einzige gedrückte Taste, deren Reihe bekannt ist. Der Inhalt von Register B wird um die Ziffer 4 mindestens einmal für eine gedrückte Taste in Reihe n inkrementiert. Die so produzierte Tastenzahl ist eine Ziffer zwischen 0 und 27. Die Schleife LPKB1 führt die Erhöhungen zu der in Register B gespeicherten Zahl aus. Die erhaltene Tastenzahl gelangt dann von Register A in Register C. Die Befehle CCF und RET führen die Steuerung zurück zum Abrufprogramm mit der Tastenzahl in Register A und C und zurückgesetztem Carry-Flag.

Subroutine KBSCAN – komplett dokumentierte Auflistung

```
KBSCAN:  LD BC,0F701 H     ; B auf F7 und C auf 01 initialisieren
                           ; C ist die Hinweisadresse zu der
                           ; geprüften Tastenreihe. B wird
                           ; schließlich benutzt, um auf die
                           ; gedrückte Taste hinzuweisen,
```

```
LPKBS:   SLA C          ; nach links verschieben, C mit
                        ; binären Nullen ausfüllen, um auf
                        ; die nächste zu prüfende Reihe
                        ; hinzuweisen;
         RET C          ; zurückgeben, wenn Reihen-Hin-
                        ; weisadresse, ein gesetztes Bit, ins
                        ; Übertrag-Flag rotiert worden
                        ; ist, d.h., wenn das Abtasten
                        ; aller Reihen abgeschlossen ist.
         LD A,C         ; Reihen-Hinweisadresse in den Akku-
                        ; mulator geben, da Ausgabe an
                        ; PB1 . . . PB7 von A erfolgt;
         CPL            ; Akkumulator ergänzen, um den Ausgang
                        ; Q1-Transistoren zu invertieren;
         CALL IO        ; Reihe prüfen, auf die das einzige
                        ; "0"-Bit im Akkumulator hinweist;
         JR Z,LBKS      ; wenn die Subroutine IO das Z-Flag
                        ; setzt, sind keine Tasten in der ge-
                        ; prüften Reihe gedrückt, daher
                        ; nächste Reihe prüfen;

STPKB:   INC B          ; wenn Subroutine IO das Z-Flag
                        ; zurücksetzt, ist eine Taste
                        ; gedrückt. Welche?

                        ; B-Register um 1 erhöhen, und zwar für
                        ; jede Stelle, um die der Akkumulator
                        ; nach links rotiert worden ist;
         RRA            ; Akkumulator nach rechts rotieren,
                        ; bis ein gesetztes Bit gefunden ist,
                        ; die Anzahl der Erhöhungen werden dem
                        ; B-Registerinhalt hinzugezählt;
         JR NC,STPKB    ; Rotation fortsetzen, bis Carry-Flag
                        ; gesetzt ist. AND 0FH prüfen, ob mehrere
                        ; Bits gesetzt sind. Falls nur eine Taste
                        ; gedrückt ist, zeigt das Ergebnis der
                        ; AND-Operation 00.
         JR NZ,CCF      ; Ist mehr als eine Taste gedrückt,
                        ; Rückkehr zum Abrufprogramm mit
                        ; gesetztem Carry-Flag veranlassen (CCF
                        ; bedeutet "Carry-Flag ergänzen", wo-
                        ; durch es zum Setzen veranlaßt wird,
                        ; da die AND-Anweisung das Carry-
                        ; Flag zurückgesetzt hat).
         LD A,C         ; Inhalt von Register C zum Akkumulator
                        ; geben (Hinweisadresse auf die Reihe
                        ; mit der gedrückten Taste).
         INC A          ; Bit D0 setzen, um den Inhibit-Eingang
                        ; (Sperr-Anschluß) beim Dekoder-IC
                        ; zu aktivieren.
```

```
            CALL IO          ; Alle anderen Reihen auf eine gedrückte
                             ; Taste prüfen.
            EX(SP), HL       ; Verzögerung von 30 Mikrosekunden
            EX(SP), HL
            EX(SP),HL
            EX(SP), HL
            IN A, (04H)      ; Dies sollte keine gesetzten Eingabe-Bits
                             ; ergeben.
            AND 0FH          ; Auf gesetzte Bits ("1"-Bits) prüfen.
            JR NZ, CCF       ; Mit gesetztem Carry-Flag zurück-
                             ; geben, wenn Tasten gedrückt sind, die
                             ; nicht zu der Reihe gehören, auf die das
                             ; C-Register hinweist.
            LD A,B           ; der aktuelle Stand von B gibt die Spalte
                             ; der gedrückten Taste an (siehe Schleife
                             ; STPKB); nun B aktualisieren, um die
                             ; Reihe der gedrückten Taste anzugeben.
                             ; Zu diesem Zweck für jede Null rechts
                             ; hinter dem "1"-Bit in Register C eine 4
                             ; hinzufügen. Für die Summierung B in
                             ; A laden.
LPKB1:      ADD A,04H        ; B war initialisiert worden, um min-
                             ; A laden.
            SRL C            ; B war initialisiert worden, um min-
                             ; destens einmal die 4 zu addieren.
            JR NC,LPKB1      ; Auf Reihen-Hinweisadresse prüfen
                             ; (gesetztes Bit)
            LD C,A           ; Wenn das Bit gesetzt ist, eine
                             ; weitere 4 addieren.
                             ; Sobald das gesetzte Bit in das Carry-
                             ; Flag rotiert worden ist, hält der
                             ; Akkumulator die Zahl (0 . . . 27) der ge-
                             ; drückten Taste fest. Diese Zahl ins
                             ; Register C laden.
CCF:        CCF              ; Carry-Flag komplementieren. Falls
                             ; LPKB1 gerade ausgeführt wurde, ist das
                             ; Carry-Flag auf logisch 1 gegangen
                             ; (gesetzt), so daß der Befehl JR NZ
                             ; unwirksam bleibt.
                             ; Daher setzt CCF das Carry-Flag zurück.
            RET              ; Mit gesetztem Carry-Flag zurück-
                             ; geben, falls keine oder mehr als eine
                             ; Taste gedrückt ist;
                             ; mit rückgesetztem Carry-Flag und der
                             ; Tastenzahl in Register A und C, falls
                             ; genau nur eine Taste gedrückt ist.
```

ANZEIGE-ROUTINEN: CONVDI UND DISPL

Die nächsten beiden Routinen, CONVDI und DISPL, benutzt das Betriebssystem des Nanocomputers®, um die zwei Reihen der sieben LEDs und die acht Sieben-Segment-Anzeigen zu betreiben. CONVDI liest einen festen Satz von 10 Speicherstellen ab (eine für jede Anzeigeeinheit) und übersetzt ihren Inhalt in eine Form, welche die richtigen LEDs bzw. Segmente aufleuchten läßt, wenn sie an das Daten-Gatter B von PIO1 gelangen. DISPL führt die eigentliche Ausgabe durch, indem sie die durch CONVDI erzeugten Bytes liest und an das Daten-Gatter B von PIO1 ausgibt. Eine typische Schrittfolge, wie sie das Betriebssystem des Nanocomputers® anwendet, um die 10 Anzeigeeinheiten zu treiben, ist folgende:

1. Die in den Speicherstellen LEDH anzuzeigenden Daten durch ADD7 + 7 bereitstellen (Weitere Einzelheiten siehe Anhang A, Beschreibung des RAM-Arbeitsbereiches).
2. CONVDI zur Durchführung der Übersetzung abrufen.
3. Schleife an der Subroutine ruft DISPL auf, bis eine Eingabe vom Tastenfeld gefunden ist.

Die im dritten Schritt genannte Schleife frischt alle 10 Anzeigen auf, dem menschlichen Auge erscheinen sie als konstant aufleuchtend.

SUBROUTINE CONVDI

Die Subroutine übersetzt vier binäre Bytes in den zugehörigen Siebensegmentcode der entsprechenden Hex-Zahl, damit die Informationen in geeigneter Form für die Anzeigeeinheit des Nanocomputers® zur Verfügung steht.

Speicherstelle

CONVDI absolute Speicherstelle – (siehe Anhang A, Tabelle A-1)

Eingabedaten

Die vier Eingabe-Bytes (die in acht hexadezimale Ziffern zu übersetzen sind) werden wie folgt gespeichert:

DATAL – LO-Daten-Byte
DATAH – HI-Daten-Byte
ADDL – LO-Adreß-Byte
ADDH – HI-Adreß-Byte
(absolute Speicherstellen siehe Anhang)

Das Zweit-Akkumulator-Register (A'), bestimmt, welche Sieben-Segment-Anzeigen aufleuchten oder dunkel bleiben:

Bit 0: logisch 1 bestimmt, die niedrigstwertige hexadezimale Ziffer in der Datenanzeige soll dunkel sein; logisch 0 bestimmt, die entsprechende hexadezimale Ziffer (die niedrigstwertigen vier Bits von DATAL) wird angezeigt.
Bei Bit 1 bis 7 ist die Funktion ähnlich. Registerpaar DE = ADDH (das letzte in der zu übersetzenden 4-Bit-Reihe) Registerpaar HL = ADD7 – 1 (eine Stelle vor der am weitesten rechts stehenden Ziffer in der anzuzeigenden Reihe der 8 Displays).

Ausgabedaten

Die Sieben-Segmentcodes für die acht hexadezimalen Anzeigeziffern werden wie folgt in den Speicherstellen ADD7 bis ADD7 + 7 gespeichert:

ADD7: – hexadezimaler Sieben-Segmentcode für die höchstwertigen vier Bits des HI-Adreß-Byte.
ADD7+1: – hexadezimaler Sieben-Segmentcode für die niedrigstwertigen 4 Bits des HI-Adreß-Byte.
ADD7+2: – hexadezimaler Sieben-Segmentcode für die höchstwertigen vier Bits des LO-Adreß-Byte.
ADD7+3: – hexadezimaler Sieben-Segmentcode für die niedrigstwertigen vier Bits des LO-Adreß-Byte.
DATA7 = ADD7+4: – hexadezimaler Sieben-Segmentcode für die höchstwertigen 4 Bits des HI-Daten-Byte.
DATA7+1 = ADD7+5: – hexadezimaler Sieben-Segmentcode für die niedrigstwertigen 4 Bits des HI-Daten-Byte
DATA7+2 = ADD7+6: – hexadezimaler Sieben-Segmentcode für die höchstwertigen 4 Bits des LO-Daten-Bytes.
DATA7+3 = ADD7+7: – hexadezimaler Sieben-Segmentcode für die niedrigstwertigen 4 Bits des LO-Daten-Byte.

Benutzte Register

Alle Register (Registerpaar DE wird wie bereits erwähnt beibehalten.)

Beschreibung

Die ersten drei Anweisungen sichern das Registerpaar DE (die Eingabe-Hinweisadresse) auf dem Stapelspeicher, tauschen den Inhalt des Registerpaars DE und HL aus (die Eingabe-Hinweisadresse ist jetzt in HL) und setzen den Akkumulator auf 0. Diese letzten beiden Maßnahmen sind für den nachfolgende Befehl RLD erforderlich, um nacheinander 4-Bit-Halbbytes zur Übersetzung in äquivalente hexadezimale Sieben-Segmentcodes in den Akkumulator zu rotieren.
Die Logik der Subroutine CONVDI basiert auf einer größeren Schleife, nämlich LOOPXX. Die Subroutine durchläuft die Schleife achtmal, während das Registerpaar von LEDL auf ADD7 + 7 erhöht wird. Vor einem letzten Test steht der Befehl INC D, um festzustellen, ob alle acht Halbbytes übersetzt sind. DE startet von LEDL aus und vergleicht im letzten Test DE mit ADD7 + 8.
Die Schleife LOOPX rotiert ein 4-Bit-Halbbyte der 4-Byte-Eingabereihe in die vier niedrigstwertigen Bits des Akkumulators. LOOPX wird viermal ausgeführt, wobei Register B als Zählwerk arbeitet. Um eine Verschiebung nach links zu erzielen, muß man den Inhalt der Speicherstellen DATAL . . . ADDH mit binären Nullen auffüllen. Die Reihenfolge der Rotation ist vom hochwertigen zum niedrigwertigen Bit, d.h. vom hochwertigen Halbbyte von ADDH zum niedrigwertigen Halbbyte von DATAL. In Bild 5-22 ist diese Operation schematisch dargestellt.
Sobald ein Halbbyte in den niedrigstwertigen vier Bits des Akkumulators ist, muß es in das hexadezimale Sieben-Segment-Äquivalent übersetzt werden. Der von den vier Bits dargestellte Wert ist aus nachstehender Tabelle ersichtlich, die mit SEGTAB bezeichnet ist:

Speicherstelle	Inhalt	aufleuchtende Segmente	Hex-Ziffer
SEGTAB	FC	a, b, c, d, e, f	0
	60	b, c	1
	DA	a, b, d, e, g	2
	F2	a, b, c, d, g	3
	66	b, c, f, g	4
	B6	a, c, d, f, g	5
	BE	a, c, d, e, f, g	6
	E0	a, b, c	7
	FE	a, b, c, d, e, f, g	8
	F6	a, b, c, f, g	9
	EE	a, b, c, e, f, g	A
	3E	c, d, e, f, g	b
	9C	a, d, e, f	C
	7A	b, c, d, e, g	d
	9E	a, d, e, f, g	E
	8E	a, e, f, g	F
	00	(keine Segmente)	Leerstelle

Die Zuordnung der Segmente in der Sieben-Segmentanzeige zu den Bits in einem 8-Bit-Byte ist wie folgt:

Segment	Bit
a	7
b	6
c	5
d	4
e	3
f	2
g	1

Das Bit 0 ist unbenutzt. Ein aufleuchtendes Segment entspricht einer logischen 1 und ein nicht aufleuchtendes einer logischen 0.
Sobald der richtige hexadezimale Sieben-Segmentcode bestimmt worden ist, wird das Register A einmal rotiert, um das nächste höherwertige Bit in das Carry-Flag zu bringen. Ein vorhandener Übertrag ersetzt den hexadezimalen Code durch Null, da ansonsten der Hex-Code zur nachfolgenden Anzeige in das entsprechende Bestimmungs-Byte übertragen wird.
Eine Wiederholung von LOOPXX findet solange statt, bis alle vier Bytes übersetzt sind. Vor der Rückführung der Steuerung zum Abrufprogramm wird der ursprüngliche Inhalt des Registerpaares DE wiederhergestellt.
Eine zusätzliche Anmerkung: Die Aufrechterhaltung der ursprünglichen 4-Byte-Eingabereihe geschieht mit Hilfe folgender Befehle:

```
PUSH AF          ; Zur Sicherstellung des in die niedrigwertige Hälfte des
                 ; Registers A rotierten Halbbytes.
POP AF           ; zur Wiederherstellung von Register A.
LD HL, DATAL     ; Bringt das höchstwertige Halbbyte der vorigen Schleife
ADD A, (HL)      ; in das niedrigstwertige Halbbyte für die nächste
LD(HL),A         ; Schleife.
```

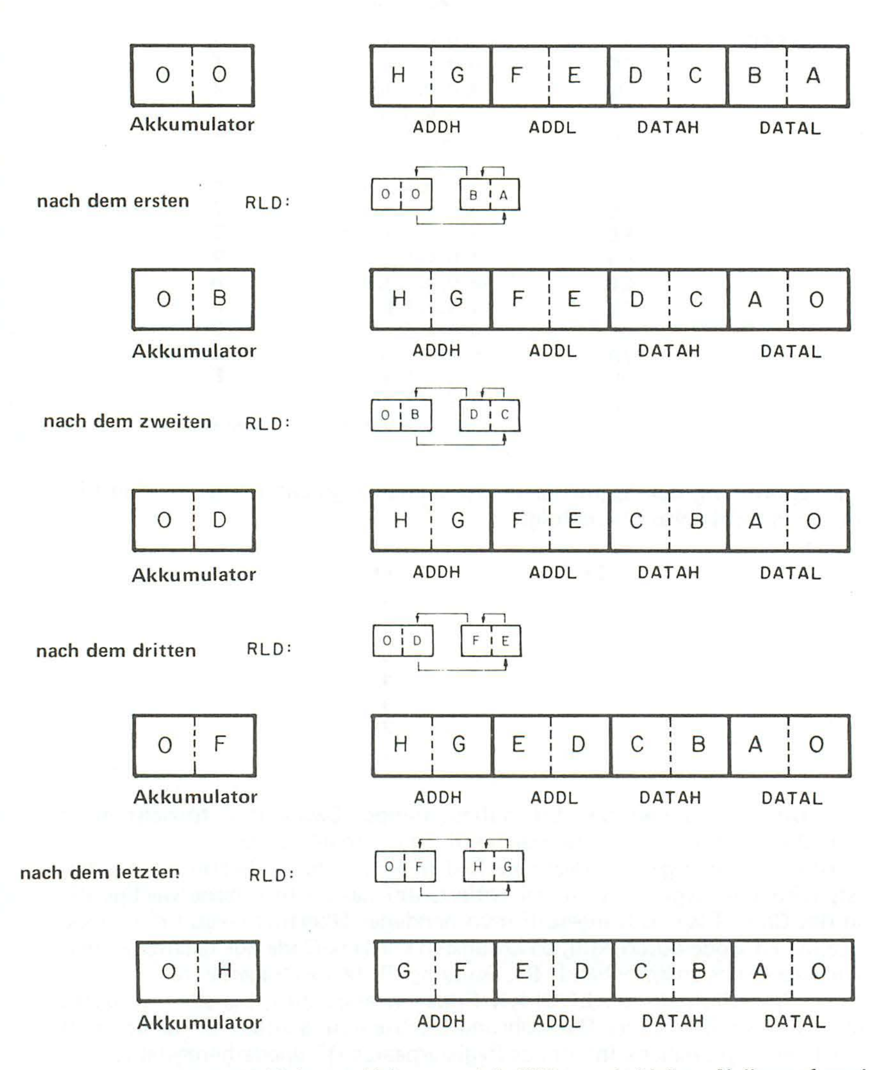


Bild 5-22. Dezimale Linksverschiebung und Auffüllung mit binären Nullen auf zwei Daten- und zwei Adreß-Anzeigeziffern.

Somit sorgen diese Befehle im wesentlichen für eine *Ringverschiebung* des Befehls RLD, weil sie das höchstwertige Halbbyte veranlassen, von der linken Seite auf die rechte Seite der Reihe zu wechseln.

SUBROUTINE DISPL

Sie hat die Aufgabe, die an den Speicherstellen ADD7, DATA7 und LEDH gespeicherte Adreß-, Daten- und Auswahlinformation anzuzeigen.

Speicherstelle

DISPL (absolute Speicherstelle – siehe Anhang A, Tabelle A-1).

Eingabedaten

- Adresse für die vier Ein-Bytecodes starten, um die vier Sieben-Segment-Anzeigen in der Adreßanzeige zu aktivieren.
- Adresse für die vier Ein-Bytecodes starten, um die vier Sieben-Segment-Anzeigen in der Datenanzeige zu aktivieren.
- Adresse für die zwei Bytes starten, deren Bits die entsprechenden Selektor-LEDs aufleuchten lassen (insgesamt 14 Dioden) (Absolute Speicherstellen – siehe Anhang).

Ausgabedaten

Die Anzeigen werden getrieben.

Benutzte Register

F, A und HL.

Gesamt-Ausführungszeit

940 ms.

Beschreibung

Der Inhalt des Registerpaars BC wird zuerst zwischengespeichert, damit diese Routine die Registerinhalte nicht ändert. Das Registerpaar HL erhält für die höherwertige Ziffer der Adreßanzeige die Adresse des Sieben-Segmentcodes. Anschließende Ausführungen der OUTLP-Schleife treiben die übrigen Anzeigen. Registers B – es tastet den Dekoder HCF4514B ab, der individuelle Anzeigen selektiv aktiviert – ist bei der hexadezimalen Ziffer 13 initialisiert. Das Register C wird mit 05, dem Gerätecode für GATTER 5, geladen. GATTER 5 ist das Datengatter für Gatter B von PIO1: Bit D0 ist die Abtasteingabe für das Dekoder/Auffang-Register HCF4514B, während die Bits D1 bis D4 entweder als Auswahleingänge zum Dekoder/Auffang-Register mit 4-zu-16 Leitungen dienen oder (zusammen mit Bits D5 und D7) die Sieben-Segment- und LED-Anzeigen auf dem Tastenfeld treiben.

Nach der Ausgabe von logischen Einsen auf jeder Datenleitung, um die Treiber-Transistoren in Q1 (BGY16) auszuschalten, tritt die Subroutine DISPL in eine kurze Verzögerungsschleife ein, die mit LP bezeichnet ist. Diese Schleife bereitet die nächste anzuzeigende Ziffer oder LED- Gruppe zur Anzeige vor. Dazu sind drei Operationen erforderlich:

a) Inhalt der Speicherstelle HL in den Akkumulator geben, b) Akkumulator komplementieren und c) Bit Null auf logisch 0 zurücksetzen.

Welche Beziehung besteht nun zwischen der Tastenfeld-/Anzeige-Hardware und der Subroutine DISPL, die mit der Anzeigen-Hardware zusammenarbeitet? Die Ziffern oder LEDs in der Anzeige leuchten auf, wenn die

entsprechenden Speicherbits logisch 1 sind. Bit D0 wird dabei nicht benutzt. Wie aus dem Schaltbild (Bild 5-21) der Tastenfeld-Anzeige ersichtlich, schaltet eine logische 0 auf eine der Datenleitungen zu dem entsprechenden Segment bzw. der LED die positive Versorgungsspannung durch. Dies ist möglich, weil die PNP-Transistoren im IC Q1 vorgespannt sind, so daß deren Kollektoren bei einer negativen Ansteuerung an der Basis auf logisch 1 gehen. Welche nun der vorgespannten Segmente oder LEDs aufleuchten, bestimmt der Dekoder HCF4514B (in Bild 5-21 Q4). Entsprechend der logischen Eingangszustände gehen die äquivalenten Ausgänge auf logisch 1 und steuern die NPN-Transistoren in Q2 und Q3 an. Die angesteuerten Transistoren schalten durch und führen am Kollektor logisch 0. Die so ausgewählten Segmente bzw. LEDs leuchten nun auf. Der Dekoder Q4 ist jedoch nur dann aktiv, wenn das Bit D0 im Zustand logisch 0 ist.
Der vorhergehende Abschnitt macht deutlich, warum die DISPL-Software den Akkumulator komplementiert und das 0-Bit zurücksetzt, bevor sie ein Anzeige-Treibbyte ausgibt. Die drei Befehle

```
OUT (C),B
DEC B
OUT (C),B
```

tasten eine neue Auswahladresse (Bits D1 . . . D4) in den Dekoder, indem sie für das Bit D0 einen Impuls erzeugen (low-high-low). Mit der gewählten richtigen Anzeige wird dann ein Byte ausgegeben, das die Ziffer oder LED-Gruppe definiert: OUT (GATTER 5), A. Register B startet von der hexadezimalen Ziffer 13 aus, wodurch zwei Erhöhungen pro angezeigtem Byte die richtige Adresse ergeben, um in das Dekoder/Auffang-Register geladen zu werden. Sind alle Anzeigen erfolgt, ist der Inhalt von Register B FF. In der Tabelle 5-6 sind die hexadezimalen Sieben-Segmentcodes für jeden Buchstaben im Alphabet, die Dezimalziffern und mehrere spezielle Symbole aufgeführt. Mit diesen Codes lassen sich in Verbindung mit der Subroutine DISPL Mitteilungen aller Art produzieren, wie z.B. bei F000 in der Laderoutine der Versuchs-Software.
Nachdem alle Bytes ausgegeben sind, sorgt die Schleife AAAA für eine kurze Verzögerung. Die nach AAAA folgenden Befehle bestimmen:

a) ob die Anzeige der Adreßziffern beendet ist (damit die Anzeige der Datenziffern beginnen kann);
b) ob die Anzeige der Datenziffern beendet ist (damit die Anzeige der Selektor-LEDs beginnen kann;
c) ob der Anzeigezyklus abgeschlossen ist (DEC B, JP P, OUTLP).

Der letzte Befehl OUT (C),B belegt alle Datenleitungen mit logisch 1. Dadurch sind während der folgenden I/O-Operation alle Anzeige-Treibertransistoren sowie der Dekoder ausgeschaltet.
Das Registerpaar BC wird dann in den ursprünglichen Zustand zurückversetzt, und die Steuerung kehrt zum Abrufprogramm zurück.

Komplett dokumentierte Auflistung – CONVDI und DISPL

```
CONVDI: PUSH DE          ; Registerpaar-Inhalt DE zwischen-
                         ; speichern, damit es in seinen Ein-
                         ; gangsstatus zurückversetzt werden
```

Tabelle 5-6. Hex-Codes zur Verwendung in der Anzeige-Routine DISPL

Buchstabe	Hex Code	Buchstabe	Hex Code
A	EE	t	IE
b	3E	U	7C
C	9C	u	38
D	F0	v	38
d	7A	W	7C, 70
E	9E	w	38, 30
F	8E	Y	4E
G	BC	Z	DA
H	6E	=	12
h	2E	—	02
I	60	_	10
i	20	°	C6
J	F0	?	CA
K	4E	\	4A
L	1C	/	26
M	EC, E0	1	0C
m	2A, 22	2	DA
N	EC	3	F2
n	2A	4	66
O	FC	5	B6
o	3A	6	BE
P	CE	7	E0
R	AC	8	FE
r	0A	9	F6
S	B6	0	FC

Die Buchstaben M und W benutzen zwei Bytes.

```
                                 ; kann, bevor es zurückkehrt.
          EX DE,HL               ; DE und HL austauschen, damit DE
                                 ; auf eine Speicherstelle hinweist, die
                                 ; vor der äußersten rechten Ziffer in
                                 ; der Reihe der acht anzuzeigenden
                                 ; übersetzten Ziffern liegt.
          XOR A                  ; Vorsorglich Akkumulator auf Null
                                 ; setzen, um den Befehl RLD benutzen
                                 ; zu können.
LOOPXX:   LD B,04H               ; Register B zählt die RLD-Befehle
                                 ; durch Rotieren der vier Bytes (siehe
                                 ; DJNZ LOOPX).
          INC DE                 ; DE aktualisieren, um auf die nächste
                                 ; Speicherstelle zum Speichern der
                                 ; übersetzten Ziffer hinzuweisen.
          LD HL,DATAL            ; HL weist jetzt auf Speicherstelle
                                 ; 0FE2, welche die niedrigstwertige
                                 ; in der rotierten 4-Byte-Reihe ist.
                                 ; Diese 4-Byte-Reihe wird RLD-
                                 ; Operationen unterworfen, während
                                 ; alle aufeinander folgenden Halb-
```

```
                ; bytes in den Akkumulator rotiert,
                ; übersetzt und gespeichert (DE)
                ; werden.
        ADD A,(HL)      ; RLD-Operation "ringverschieben"
                ; durch Addition des Akkumulators,
                ; dessen linkes Halbbyte 0 ist, und
                ; zwar  zu dem niedrigstwertigen Byte
                ; der rotierten 4-Byte-Reihe, deren
                ; rechtes Halbbyte mit Nullen aufge-
                ; füllt worden ist. Als Folge wird das
                ; rechte Halbbyte im Akkumulator
                ; (welches das höchstwertige Halbbyte
                ; in der 4-Byte-Reihe WAR – vor dem
                ; letzten RLD-Befehl) das niedrigst-
                ; wertige Halbbyte.
                ; (siehe Abbildung in Dokumentation
                ; über CONVDI)
        LD (HL),A       ; Das niedrigstwertige Byte aktuali-
        LD A,E          ; sieren.Prüfen, ob letztes Halbbyte
        CP 0C2H         ; übersetzt worden ist durch Fest-
                ; stellen, ob DE auf das letzte Byte
                ; in der 8-Byte-Reihe hinweist, das
                ; übersetzt werden soll.
        JR Z,KBINI      ; Falls ja, springen, um DE und RET
                ; umzuspeichern.
        XOR A           ; A vor der RLD-Verschiebung der 4-
                ; Bytes erneut initialisieren.
LOOPX:  RLD             ; Linke Dezimalstelle rotieren.
        INC HL          ; Durch Erhöhung der Wertigkeit auf
                ; nächstes Byte hinweisen.
        DJNZ LOOPX      ; 4 mal rotieren.
        PUSH AF         ; Akku-Inhalt zwischenspeichern.
        LD HL SEGTAB    ; Halbbyte in unterer Hälfte des Akku-
        ADD A,L         ; mulators übersetzen, indem es als
        LD L,A          ; Index in der Sieben-Segment-Tabelle
                ; benutzt wird.
        LD A,(HL)       ; Bytes anzeigen.
        LD C,A          ; Übersetztes Halbbyte in Register C
                ; zwischenspeichern.
        EX AF,AF'       ; Durch Prüfung des A'-Registers fest-
                ; stellen, ob dieses Byte angezeigt
                ; werden sollte.
        RLC A           ; Falls entsprechendes Bit in A' gesetzt
                , ist, darf das Byte nicht angezeigt
                ; werden, sondern mit Nullen auf-
                ; füllen.
        JR NC, NOBLXX   ; Nicht mit Nullen auffüllen, falls das
                ; Carry-Flag rückgesetzt ist.
        LD C,00H        ; Register C mit Nullen auffüllen, falls
                ; Carry-Flag gesetzt ist.
```

```
NOBLXX: EX AF,AF'          ; Wieder austauschen.
        LD A,C             ; Übersetztes Halbbyte zu A über-
                           ; tragen.
        LD (DE),A          ; Übersetztes Byte zur anschließenden
                           ; Anzeige durch Subroutine DISPL
                           ; speichern.
        POP AF             ; Zur Vorbereitung für "Ringverschie-
                           ; bung" der RLD-Operation Halbbyte
                           ; vor Übersetzung umspeichern.
        JR LOOPXX          ; Zurückspringen und mit Übersetzung
                           ; des nächsten Bytes beginnen.
KBINI:  POP DE             ; DE umspeichern.
        RET
-----------------------------------------------------------------
DISPL:  PUSH BC            ; Inhalt des Registerpaars BC auf
                           ; Stapelspeicher zwischenspeichern,
                           ; damit die Umspeicherung vom Ein-
                           ; gangsstatus möglich ist.
        LDHL,DATA7-1       ; Adresse DATA7 – 1 = 0FBD ins
                           ; Registerpaar HL laden. DATA7 – 1
                           ; entspricht DATA7 + 3, was die
                           ; Adresse der höchstwertigen Adressen-
                           ; Anzeigeziffer ist (übersetzt durch
                           ; CONVDI).
        LD BC,1300H+05H    ; 13 in B und 05 in C laden. 05 ist die
                           ; Gatteradresse für Gatter B von PIO1,
                           ; dem Treibergatter der Anzeigen.
                           ; Der Inhalt von Register B wird zum
                           ; Dekoder ausgegeben, um die laufende
                           ; Anzeigeeinheit auszuwählen.

; ÜBEREINSTIMMUNG ZWISCHEN DEN WERTEN DES REGISTERS B,
; DER AUSGEWÄHLTEN ANZEIGE UND DER ANGEZEIGTEN
; SPEICHERADRESSE
; Jede OUTLP-Ausführung führt zu zwei Erhöhungen im Register B

; Register B      ausgewählte Anzeige               angezeigte
                                                    Speicheradresse
; 11 oder 10      ADD1                              ADD7+2
; 0F oder 0E      ADD2                              ADD7+1
; 0D oder 0C      ADD3                              ADD7
; 13 oder 12      ADD0                              ADD7+3
; 0B oder 0A      DATA0                             DATA7+3
; 09 oder 08      DATA1                             DATA7+2
; 07 oder 06      DATA2                             DATA7+1
; 05 oder 04      DATA3                             DATA7
; 03 oder 02      IY, IX, HL, DE, BC, AF, IR        LEDH+1
; 01 oder 00      ARS, ERR, SP, PC, MEM, I/O, BRK   LEDH
```

ADD0 . . . ADD3 sind die vier Adressanzeigen, die von links nach rechts aufgeführt sind. DATA0 . . . DATA3 = die vier von links nach rechts aufgeführten Datenanzeigen.

```
OUTLP:   LD A,FFH        ; Der Sperreingang des ICs HCF4514B
                         ; ist aktiviert, weil D0 high ist.
         LD A,03H        ; Verzögerungsschleife von 19,2 ms.
LP:      DEC A
         JR NZ,LP
         LD A,(HL)       ; Erstes anzuzeigende Byte in A laden.
         CPL             ; Wegen Invertierung des Charakters
                         ; der Q1-Transistoren Auukumulator
                         ; auffüllen.
         RES 0,A         ; Bit D0 rücksetzen, damit Sperrein-
                         ; gang des ICs HCF4514B inaktiv wird.
         OUT(C),B        ; Das Anzeige-Wählbyte aus Register B
                         ; in das Auffang-Register des ICs
                         ; HCF4514B übertragen.
         DEC B           ; Erstens Decrement (Verminderung)
         OUT (C),B       ; von Register B.
         OUT(05H),A      ; Anzeigebyte an PB1 . . . PB7 aus-
                         ; geben.
         LD A,10H        ; Noch eine kurze Verzögerungs-
                         ; schleife.
AAAA:    DEC A
         JR NZ,AAAA
         DEC HL          ; HL aktualisieren, um auf das nächste
                         ; Byte für die Anzeige hinzuweisen.
         LD A,L          ; Prüfen, ob alle ADDR-Bytes
         CP 0B9H         ; angezeigt worden sind.
         JR NZ,NXT1      ; Falls nicht, zur Anzeige zurück und
                         ; nächstes ADDR-Byte anzeigen.
         LD L,0C1H       ; Falls ja, L initialisieren, um DATEN-
                         ; Bytes anzuzeigen.
NXT1:    CP 0BDH         ; Prüfen, ob alle DATEN-Bytes ange-
         JR NZ,NXT2      ; zeigt worden sind.
         LD L,0B9H       ; Falls ja, L zwecks Anzeige der LED-
                         ; Bytes initialisieren.
NXT2:    DEC B           ; Falls nicht, fortfahren. Zweites
                         ; Decrement von Register B.
         JP P,OUTLP      ; Solange B = 0 oder größer ist, weiter
                         ; anzeigen.
         OUT (C),B       ; Sobald B negativ wird, d.h. auf FF
                         ; geht, für Bit D0 eine logische 1
                         ; ausgeben, um das IC HCF4514B
                         ; zu sperren.
         POP BC          ; Inhalt des BC-Registerpaars umspei-
                         ; chern.
         RET
```

Serien-I/O-Routine

Bis zu diesem Punkt tauschen alle I/O-Schaltungen die Datenbytes mit der Z-80-CPU durch *parallele* Bit-Übertragung. Das heißt, die zu oder von der CPU übertragenen acht Bits werden über acht getrennte Leitungen übertragen, eine Leitung für jedes Bit.

Dies ist für räumlich dicht beieinander stehende Geräte eine ausgezeichnete Kommunikationsmethode. Aber bei Geräten, die mehrere hundert Meter oder sogar Kilometer auseinander liegen, wären die Kosten für die acht Parallel-Leitungen nicht zu verantworten. Die kosteneffektive Alternative zur Parallel-Kommunikation über weitere Entfernungen ist die Serien-I/O.

Das Serien-Interfacing zur Z-80-CPU erfüllt folgende Funktionen:

SERIEN-EINGANG:
1. Entgegennahme der Seriendaten für das externe Gerät, Bit für Bit.
2. Wiederherstellung des 8-Bit Wortes.
3. Anlegen des 8-Bit Wortes in das interne CPU-Register, das von der Software als Bestimmungsregister gewählt wird.

SERIEN-AUSGANG:
1. Empfang des 8-Bit Wortes zwecks Ausgabe zum externen Gerät.
2. Ausgabe des Byte zum externen Gerät, Bit für Bit.

In beiden Fällen nimmt die Z-80-CPU über *parellele* 8-Bit-Bytes Verbindung mit der Serien-/I/O-Schaltung auf. Das Serien-Interfacing hat die Funktion, die Übersetzung von Serien- zum Parallelsystem oder umgekehrt vorzunehmen und das Anlegen oder Lesen von Daten auf dem Kommunikationsgerät zeitlich abzustimmen.

Bei in den parallelen I/O-Schaltungen, die in Kapitel 4 beschrieben sind, dienen Synchronisations-Impulse (IN XX und OUT XX) zur Koordination der Datenanlegung auf dem Kommunikationsgerät (D0 . . . D7) durch den Sender bzw. zum Ablesen der Daten vom Gerät durch den Empfänger. Diese Aufgabe wird von separaten Leitungen, den *Steuerleitungen,* wahrgenommen. Für die Serien-Kommunikation ist eine ähnliche Koordination erforderlich, aber für diese Aufgabe stehen keine Extra-Leitungen zur Verfügung. (Einige Serienglieder erhalten mehr als nur den Mindestsatz von Leitungen: Eingang, Ausgang, Masse und Netz). Um die Aktitäten von Empfänger und Sender zu koordinieren, sind die folgenden Absprachen notwendig:

1. Sowohl Empfänger als auch Sender einigen sich, die Leitung in einem bestimmten Rhythmus pro Sekunde abzutasten. Diese Abtastgeschwindigkeit wird Schrittgeschwindigkeit in BAUD genannt und hat für Empfänger und Sender dieselbe Frequenz.
2. Empfänger und Sender einigen sich: jedes gesendete Byte setzt sich aus einer Folge von wechselnden Spannungspegeln auf der Sendeleitung für den Sender und auf der Empfangsleitung für dem Empfänger zusammen. Da eine Masseleitung beide verbindet, benutzen beide denselben Bezugspunkt zur Bestimmung des Leitungspegels. In der internationalen Literatur gilt in der Regel für die Logikpegel 0 und 1 in Beziehung zu den entsprechenden Spannungspegeln folgende Schreibweise:

   Spannung $\leqslant H_L$, das abgetastete Bit befindet sich im Zustand logisch 0.

Spannung ⩾ V_H, das abgetastete Bit befindet sich im Zustand logisch 1.
3. Empfänger und Sender einigen sich: jedem Byte (oder 8-Bit-Strom) geht ein Low-Bit voraus, das START-Bit. Es zeigt die Übertragung eines neuen Bytes an. Zwei STOP-Bytes (logisch 1) schließen die Byte-Übertragung ab.

Von den genannten Absprachen 1, 2 und 3 gibt es viele Abwandlungen, besonders in den Assoziationen logischer Pegel – Spannungspegel und der Anzahl der Start- und Stop-Bits. Wie bereits festgestellt, sind die Absprachen für das Serien-Interfacing des Nanocomputers® zutreffend.
Bild 5-23 zeigt, wie logische Pegel auf einem Serien-Verknüpfungsglied zur Übertragung des hexadezimalen Bytes AC aussehen würde. Beachten Sie, daß der Ruhezustand der Leitung nach vollzogener Byte-Übertragung high ist. Wenn die Leitung auf low geht, signalisiert sie die Übertragung eines Start-Bits vor den Daten-Bits.

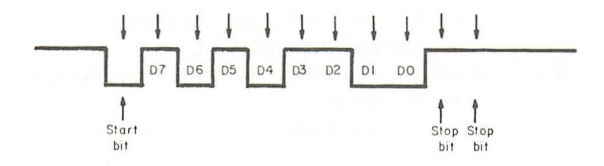


Bild 5-23. Serien-Übertragung eines 7-Bit-Zeichen mit einem Start- und zwei Stop-Bits. Bei jedem Bit (↓) tastet der Empfänger die Leitung ab.

Die Parallel- zur Serien- und Serien- zur Parallel-Umsetzerfunktion in einem Serien-Interface läßt sich sowohl in Hardware als auch in Software realisieren. Mehrere Halbleiter-Hersteller bringen integrierte Schaltungen auf den Markt, die als Umsetzer in beide Richtungen arbeiten. Sie erfüllen außerdem noch andere Funktionen, die Übertragungsfehler erkennen und melden. Die integrierten Schaltungen werden UARTS (Universal Asynchronous Receiver/Transmitters – Universeller Asynchron-Empfänger/Sender) oder USARTS (Universal Synchronous Receiver/Transmitter – Universeller Synchron-Empfänger/Sender) genannt. Die PC-Platine des Nanocomputers® läßt sich mit einem einfachen Ergänzungssatz auf USART erweitern.
Da die PC-Platine des Standard-Nanocomputers® ohne USART gestaltet ist, realisiert das Betriebssystem die Parallel/Serien- und die Serien/Parallel-Umsetzung Funktion über zwei Subroutinen TTYI1 für die Serieneingabe und TTY0 für die Serienausgabe.

Beide Routinen benutzen den Gatter A-Daten-BUS des PIO1 zur Kommunikation mit den Peripheriegeräten, die an den Nanocomputer® angeschlossen sind. Dies kann über die Steckerleiste J13 der Kassettenrekorder oder über die Steckerleiste J5 die Stromschleife RS322-C, 20 mA bzw. TTL-Geräte sein. Wie bereits erwähnt, ist PIO1 von dem Betriebssystem programmiert, um nach der Steuermethode zu arbeiten, wobei die Daten-Bits für GATTER A wie folgt definiert sind:

10001111

Daraus geht hervor, daß die Leitungen PA7, PA3, PA2, PA1 und PA0 für die Eingabe und PA6, PA5 und PA4 für die Ausgabe reserviert sind. Die Routine TTYI1 empfängt die Serieneingabe über Leitung PA7 und die Routine TTYO gibt die Seriendaten über die Leitung PA4 aus. Im Schaltbild 5-6 ist PA4 mit TDX (transmit data – übertragene Daten) und PA7 mit RDX (received data – empfangene Daten) bezeichnet.

SUBROUTINE TTYI1

Sie hat die Aufgabe, ein seriell zur CPU übertragenes 7-Bit-Zeichen in den Akkumulator und das C-Register einzugeben. Für die Byte-Übertragung ist selbstverständlich ein Start- und zwei Stop-Bits vorausgesetzt.

Speicherstelle: TTYI1

(Absolute Speicherstelle – siehe Anhang A, Tabelle A-1)

Eingabedaten:

Keine.

Ausgabedaten

Das Eingabezeichen wird in die Register C und A geladen.

Benutzte Register

A, B und C.

BESCHREIBUNG

Die erste Aufgabe ist das Warten auf ein Start-Bit (logisch 0). Dies bewirken die ersten drei Programm-Anweisungen. Das Eingangs-Gatter – Gerätcode 04 – wird solange abgelesen, bis D7 auf logisch 0 geht. Das Carry-Flag zeigt dies durch Rücksetzen an, nachdem der Befehl RLA D7 ins Carry-Flag rotiert ist.
Die Routine BAUDHF überprüft, ob tatsächlich ein Start-Bit auf der Leitung liegt. BAUDHF ist eine Verzögerungsschleife, deren Dauer genau der halben Abtast-Periodenzeit entspricht. Am Ende der Verzögerungszeit wird Gatter 04 erneut abgetastet. Dadurch stellt man fest, ob Leitung D7 noch low ist. Falls D7 nicht mehr logisch 0 ist, handelte es sich um kein echtes Start-Bit. Die Steuerung springt an den Anfang der Routine zurück. Ist im anderen Fall D7 noch logisch 0, handelt es sich um ein echtes Start-Bit. Als nächstes sind also die acht Daten-Bits einzugeben.
Der Befehl initialisiert das Register B als Zähler. Warum 9 anstatt 8? Weil das erste abgetastete Bit das Start-Bit ist. Um das Start-Bit und alle acht Bit einzuschließen, läuft LOOPTI neunmal durch. Zuerst geht das Bit in D7 des Akkumulators ein. Der Befehl RLA kopiert Bit D7 ins Carry-Flag. Der Befehl RR C rotiert die Bits im Register C eine Stelle nach rechts, wobei sich D7 mit dem Inhalt des Carry-Flag füllt. So rotieren alle Eingabe-Bits eins nach dem anderen ins C-Register, wobei die niedrigstwertigen Bits zuerst ankommen und schließlich an der richtigen Stelle im umgewandelten Byte landen. Nach Eingabe der acht Daten-Bits ist der

nächste Schritt die Prüfung auf Stop-Bits. Dazu sind folgende Befehle notwendig:

```
INC B
IN A, (PORT 0)
RLA
JR NC, LOOPTI
```

Ist keine Carry-Funktion vorhanden, ist das nächste zu übertragende Bit low, also kein Stop-Bit. In diesem Falle geht die Steuerung nach Initialisierung des B-Registers auf 1 an LOOPTI zurück, um erneut nach einem Stop-Bit zu suchen. Der Versuch wiederholt sich solange, bis ein Stop-Bit gefunden ist. Erst dann übernimmt die Abruf-Routine die Steuerung, wobei Register A und C das Eingabe-Byte enthalten. Das höchstwertige Bit wird zurückgesetzt! Die zusätzliche Vereinbarung findet häufig bei der seriellen Kommunikation ihre Anwendung. Dabei ist das achte Bit der Paritäts-Indikator. In vielen Systemen ist das achte Bit gesetzt, wenn die Parität der sieben anderen Bits gerade ist (z.B. wenn 2, 4 oder 6 Bits logisch 1 sind), anderenfalls ist das Paritätsbit zurückgesetzt. Diese Vereinbarung erleichtert die Fehlersuche bei der Übertragung von Daten. Der Empfänger prüft, ob der Paritäts-Indikator mit dem Wert der sieben anderen Bits übereinstimmt. Wie bereits erwähnt, setzt TTYI1 das achte Bit D7 immer zurück, so daß beim Serien-I/O-Interface des Nanocomputers® keine Paritätsprüfung stattfindet.

SUBROUTINE TTYO

Sie hat die Aufgabe, ein Einzelbyte aus dem C-Register an ein Peripheriegerät seriell auszugeben. Das zu übertragende Byte wird mit einem Start- und zwei Stop-Bits ergänzt.

Speicherstelle

TTYO (absolute Speicherstelle – siehe Anhang A, Tabelle A-1)

Eingabedaten

Das C-Register muß die zu übertragenden 8-Bit-Zeichen enthalten.

Ausgabedaten

Der Inhalt des C-Registers wird in 11 aufeinanderfolgende Bits (ein Start-, acht Daten- und zwei Stop-Bits) mit Hilfe eines Übertragungsgliedes seriell übertragen.

Benutzte Register

A und F

Beschreibung

Damit beim Ablauf der Subroutine der ursprüngliche Inhalt des Registerpaares BC erhalten bleibt, erfolgt zuerst eine Zwischenspeicherung in den Stapelspeicher. Dies ist notwendig, damit nach Ablauf der Routine und Rückkehr zum Abrufprogramm der ursprüngliche Wert im Register BC für

die weitere Verarbeitung zur Verfügung steht. Der Zweck des Befehls AND A ist die Nullstellung des Carry-Flags. Das auf Null gestellte Carry-Flag dient schließlich als Start-Bit. Das B-Register wird auf dezimal 11 initialisiert und dazu benutzt, die Bits bei der Übertragung zu zählen. Der Übertragungsvorgang ist beendet, wenn alle 11 Bits verarbeitet sind. Die Schleife LOOPTT wird insgesamt 11 mal wiederholt.
Der erste Befehl in LOOPTT ist, anders als in einer OUTPUT-Routine erwartet, ein INPUT-Befehl. Der Grund für diesen Befehl ist herauszufinden, wie der momentane logische Zustand auf allen zum Gatter 04 gehörenden Leitungen ist, die bei der seriellen Übertragung unbenutzt bleiben. Das sind alle Leitungen mit Ausnahme von D4. Der folgende Befehl ändert nur D4, daher bleibt der Zustand der anderen Leitungen unverändert. Die ersten drei der folgenden Befehle

```
RES 4,A
JR NC, NXTTT
SET 4,A
NXT: OUT (04H),A
```

setzen Bit D4, um den Inhalt des Carry-Flags wiederzugeben. Da das Carry-Flag vom Befehl AND A zurückgesetzt ist, wird das Start-Bit (low) als erstes übertragen. Der OUT-Befehl legt das Ausgabe-Bit auf die Leitung D4. Eine Abruf-Routine BAUD verursacht eine Verzögerung von genau einer Abtastperiode. Dadurch ist die zeitliche Abstimmung zwischen den Ausgabe-Bits genauso, wie vom Empfänger erwartet. Der Befehl SCF setzt das Carry-Flag, so daß der RR C-Befehl eine logische 1 in Bit D7 des C-Registers rotiert. Dies geschieht zur Vorbereitung für die Übertragung von zwei Stop-Bits, nachdem die acht Daten-Bits gesendet sind. Außerdem rotiert der Befehl RR C das nächste zu übertragende Bit ins Carry-Flag. Ist der Befehl DJNZ LOOPTT ausgeführt, definiert das Carry-Flag exakt das Setzen des Bits D4 vor der Übertragung. Nachdem neun Bits übertragen sind, werden die ursprünglich durch den SCF-Befehl gesetzten Bits ins Carry-Flag rotiert und als zwei Stop-Bits übertragen.
Sobald die Übertragung der 11 Bits abgeschlossen ist, übernimmt das BC-Register wieder aus dem Stapelspeicher den ursprünglichen Inhalt; die Steuerung geht an die Abruf-Routine zurück.

Komplett dokumentierte Auflistung – TTYI1 und TTYO

```
TTYI1:   IN A,(04H)      ; Auf ein Start-Bit im Zustand low
                         ; warten.
         RLA             ; D7 ins Carry-Flag rotieren.
         JR C,TTYI1      ; Falls D7 nicht low ist, weiter warten.
         CALL BAUDHF     ; Falls D7 low ist, feststellen, ob es
                         ; wirklich ein Start-Bit ist. Die Hälfte
                         ; einer Abtastperiode warten.
         IN A,(04H)      ; Nachsehen, wie D7 jetzt aussieht.
         RLA             ; D7 ins Carry-Flag rotieren.
         JR C,TTYI1      ; Falls D7 nicht low ist, zurückgehen in
                         ; Wartestellung bei TTYI1.
         LD B,09H        ; B als Zählregister vorbereiten.
LOOPTI:  IN A,(04H)      ; Ein Daten-Bit eingeben (beim ersten
```

```
                                ; Mal ist dies immer noch das Start-Bit).
         RLA                    ; D7 (das Eingabe-Bit) ins Carry-Flag
                                ; rotieren.
         RR C                   ; Carry-Flag-Inhalt (das Eingabe-Bit) in
                                ; D7 von Register C rotieren.
         CALL BAUD              ; Genau eine Abtastperiode warten.
         DJNZ LOOPTI            ; Bit-Zähler B erniedrigen. Falls B nicht
                                ; 0 ist, das nächste Bit nehmen.
         INC B                  ; Falls B = 0, sind alle Bits erhalten.
                                ; B für eine mögliche Schleife an
                                ; LOOPTI einrichten.
         IN A,(04H)             ; Eingeben, was ein Stop-Bit sein sollte.
         RLA                    ; Ist es eins?
         JR NC,LOOPTI           ; Es ist keins, falls das Carry-Flag
                                ; rückgesetzt ist. Zur Dateneingabe zu-
                                ; rückspringen. Schleife bilden, damit
                                ; die letzten acht Bits übertragen
                                ; werden, bevor die Stop-Bits in Register
                                ; C sind.
         LD A,C                 ; Falls Carry-Flag gesetzt, ist es ein
                                ; Stop-Bit. Eingabe-Byte zum Akku-
                                ; mulator übertragen.
         AND 7FH                ; Höchstwertiges Bit rücksetzen.
                                ; Keine Paritätsprüfung
         LD C,A                 ; Eingabe-Byte ins C-Register laden.
         RET
-----------------------------------------------------------------------
TTYO:    PUSH BC                ; Inhalt des Registerpaars BC zwischen-
                                ; speichern.
         AND A                  ; Carry-Flag zurücksetzen.
         LD B,0BH               ; Dezimal 11 ins B-Register laden, das
                                ; als Bit-Zähler arbeitet.
LOOPTT:  IN A,(04H)             ; Zustand der zum Datengatter A von
                                ; PIO1 gehörenden Bits lesen.
         RES 4,A                ; Carry-Flag-Inhalt ins Bit D4 kopieren.
         JR NC,NXTT             ;
         SET 4,A
NXTT:    OUT (04H),A            ; Bit D4 ausgeben, ohne den Zustand
                                ; der anderen Bits bei Gatter 04 zu
                                ; ändern.
         CALL BAUD              ; Genau eine Abtastperiode verzögern.
         SCF                    ; Carry-Flag setzen, damit High-Bits
                                ; in C rotiert werden, um als Stop-Bits
                                ; zur Verfügung zu stehen.
         RR C                   ; High-Bit ins höchstwertige Bit von
                                ; Register C rotieren. Nächstes Bit zur
                                ; Übertragung ins Carry-Flag rotieren.
         DJNZ LOOPTT            ; B erniedrigen. Falls es nicht 0 ist,
                                ; zurückgehen und das nächste Bit aus-
                                ; geben.
```

```
POP BC                  ; Falls B = 0 ist, sind alle zu einem 8-Bit-
                        ; Wort gehörigen 11 Bits übertragen.
                        ; Inhalt des B-Registerpaares umspei-
                        ; chern.
RET
```

Die nachfolgenden Versuche sind dazu bestimmt, Sie mit der Tastenfeld-Anzeigeeinheit des Nanocomputers® sowie der zugehörenden System-Software vertraut zu machen.

Versuch Nr.	Bemerkung
1	Macht Sie mit der Anzeige-Software bekannt, die in der Lage ist, Anzeigedaten und die Stellung des angezeigten hexadezimalen Bytes auszuwählen.
2	Zeigt Ihnen, wie Sie die System-Software des Nanocomputers® zum Lesen der Tastenfeld-Eingabe und Betreiben der Anzeigen einsetzen. Die Beschreibung der in diesem Versuch verdeutlichten Techniken folgt eingehend in den Kapiteln 6 und 7.

VERSUCH NR. 1

Der Versuch zeigt Ihnen, wie die LEDs und Sieben-Segment-Anzeigen auf der Tastenfeld/Anzeigeeinheit des Nanocomputers® betrieben werden. Es ist ein Anzeige-Testprogramm, das alle möglichen Bit-Muster sequenziell in allen Anzeigestellungen anzeigt. Durch Veränderung der Verzögerungsschleifendauer zwischen den Stellungswechseln können Sie den Eindruck erwecken, als ob man alle Anzeigeziffern gleichzeitig betreibt.

Programm DDRIVE

```
Objekt-Code            Quell-Code        Bemerkung

                       NAME DDRIVE
010500   DDRIVE:       LD BC,0005H       ; B enthält anzuzeigende Daten. C enthält
                                         ; Gerätecode für Ausgangsgatter (PIO1,
                                         ; B-Datengatter).
3E00                   LD A,PSEL         ; A enthält den Anzeige-Stellenwähler.
00                     NOP               ; Füllwort, damit dieses Programm in das
                                         ; nächste Programm hineinpaßt, ohne die
                                         ; meisten der Bytes umladen zu müssen.
ED79                   OUT (C),A         ; Anzeigeadresse an HCF4514 durch
                                         ; Schalten von D0 ausgeben.
3C                     INC A
ED79                   OUT (C), A
3D                     DEC A
ED79                   OUT (C),A
ED41                   OUT (C), B        ; Ausgang-Daten
76                     HALT
```

1. Schritt

Programm laden, beginnend bei Speicherstelle DDRIVE. Der erste Befehl

LD BC,0005H initialisiert das C-Register zur Aufnahme der Adresse des Datengatters von PIO1, B-Datengatter. Register B wird mit den Daten initialisiert, die an Gatter 05 auszugeben sind. Der zweite Befehl LD A, PSEL lädt die Adresse der Reihe von sieben LEDs oder Sieben-Segment-Anzeigen in den Akkumulator, welche die Datenausgabe an Gatter 05 zeigen soll. Sind die Daten und die Anzeigestellung einmal bestimmt, wird das Daten-Byte in Register B über das IC HCF4514 zur Anzeige ausgegeben.

Führen Sie das geladene Programm aus, indem Sie das Byte 00 für die Daten und die Anzeigestellung benutzen. Das sind die bereits im Programm geladenen folgenden zwei Befehle:

LD BC, 0005H zum Setzen der Anzeigedaten und Auswählen des Ausgangsgatters (05H).

LD A,00H zum Setzen der Anzeigestellung.

Alle Sieben-Segment-Anzeigen gehen aus, während die folgenden LEDs aufleuchten:

BRK, SP, PC, MEM, I/O, ERR und ARS.

Bei den LEDs handelt es sich um die LED-Reihe, die an dem S0-Ausgang des ICs HCF4514B angeschlossen ist. Da der S0-Ausgang aktiv wird, wenn die vier Dateneingänge zum HCF4514B logisch 0 sind, ist das Anzeigestellungs-Byte 00. Welche Bedeutung hat das Daten-Byte in Register B? Es übt zwei Funktionen aus. Erstens gibt das Signal der D0-Leitung die Ausgänge des Dekoders HCF4514B frei oder es blockiert sie. Da D0 im Daten-Byte 00 low ist (und nur immer dann), gibt die Ausgabe von 00 an Gatter 05 die Ausgänge des Dekoders frei. Die Bits D1 . . . D7 bestimmen, welche LEDs aufleuchten.

Werden im IC Q1 (Bild 5-21) der Tastenfeld-Anzeige alle Transistoren mit logisch 0 angesteuert, schalten sie durch. Bei einer Ansteuerung mit logisch 1 sind sie gesperrt. Daher läßt das Daten-Byte 00 alle LEDs in der ausgewählten Anzeigestellung aufleuchten.

2. Schritt

Unter Beibehaltung der Anzeigestellung 00 das Daten-Byte am Speicherplatz DRIVE + 2 in 01 ändern. Wie reagiert die Anzeige? Führen Sie das Programm aus, um festzustellen, ob Ihre Vermutung richtig war.

Alle Anzeigen verlöschen. Der Daten-Byte-Ausgang zum Daten-Gatter B von PIO1 versetzte D0 in den Zustand logisch 1. Der Sperreingang (Pin 1) vom Dekoder HCF4514 ist über PB0 mit D0 verbunden und HIGH-aktiv. Das niederwertigste Bit D0 vom Daten-Byte sperrt also im Zustand logisch 1 alle Ausgänge des Dekoders. Folglich ist keine Anzeigestellung aktiviert, so daß alle Anzeigen dunkel bleiben.

3. Schritt

Die folgende Tabelle führt einige interessante Daten und Positions-Bytes auf mit einer Beschreibung der beobachteten Anzeigen. Vergleichen Sie, ob die Tabelle mit Ihren eigenen Beobachtungen übereinstimmt.

Daten (DDRIVE + 2)	Stellung (DDRIVE + 4) = PSEL	Anzeigebeschreibung
00	00 und 20	BRK, SP, PC, MEM, I/O, ERR, ARS leuchten auf, alle Sieben-Segment-Anzeigen sind dunkel. Dieselbe Anzeige wird durch die Positions-Bytes 00 und 20 erzeugt, weil die Bits D1 bis D4 die einzigen Bits sind, die für den Chip HCF4514B Eingabe sind. Daher wählen 00, 20, 40, 60, 80, A0, C0 und E0 dieselbe Anzeigestellung.
01	jedes Byte	Alle Anzeigen aus.
00	01	BRK, SP, PC, MEM, I/O, ERR, ARS alle an. Alle 7-Segment-Anzeigen aus. Dies ist dasselbe, als ob beide Daten- und Positions-Bytes 00 wären, weil die Anzeigestellung durch die Bits D1 bis D4 bestimmt wird.
00	02	IY, IR, AF, BC, DE, HL, IX an. Alle 7-Segment-Anzeigen aus.

Daten	Stellung	7	6	5	4	3	2	1	0	
00	04					8				
00	06						8			
00	08							8		
00	0A								8	
00	0C	8								
00	0E		8							
00	10			8						
00	12				8					
00 00	14, 16, 18 1A, 1C, 1E	Alle Anzeigen dunkel								
82	04					U				(Bit D7 entspricht Segment a)
02	04					0				(Bit D1 entspricht Segment g)

Programm DISTST

```
Objekt-Code            Quell-Code          Bemerkung
                       NAME DISTST
010500    DISTST:      LD BC,0005H         ; B enthält anzuzeigende Daten.
                                           ; C enthält Ausgabe-Gerätecode.
AF        DATLP:       XOR A               ; A enthält anzuzeigende Stellung.
160A                   LD D,0AH            ; D ist der Anzeige-Stellungszähler.
ED79      OUTPUT:      OUT (C),A           ; Anzeigeadresse durch Schalten von
                                           ; Bit D0 an HCF4514B ausgeben.
3C                     INC A
ED79                   OUT (C),A
3D                     DEC A
ED79                   OUT (C),A
ED41                   OUT (C),B           ; Ausgabedaten
3C                     INC A               ; Positions-Hinweiseadresse erhöhen,
                                           ; damit sie auf die nächste Anzeige-
                                           stellung hinweist.
```

```
3C              INC A
CDE301          CALL DELAY     ; Unterbrechen, damit Anzeige eine
                               ; kurze Zeitspanne lang konstant ist.
15              DEC D          ; Positionszähler vermindern.
20EE            JR NZ,OUTPUT   ; falls D nicht 0 ist,
                               ; um Byte zur nächsten Anzeigen-
                                 stellung auszugeben.
04              INC B          ; Falls alle Anzeigestellungen ge-
                               ; prüft sind, Ausgabedaten aktuali-
                               ; sieren.
04              INC B            nochmals starten mit neuem Daten-
18E7            JR DATALP      ; Byte.
D5      DELAY:  PUSH DE        ; DE zwischenspeichern.
16F0            LD D,0F0H      ; Zeit-Byte
CDF2F9  DREGL:  CALL BAUD      ; BAUD ist eine Routine im Betriebs-
                               ; system, die genau eine Abtastperiode
                               ; verzögert. Die Länge der Periode wird
                               ; über ein Zeit-Byte gesetzt, das im
                               ; Speicher ist. In der Subroutine
                               ; DELAY beträgt die Verzögerung 16
                               ; (Basis 10) Abtastperioden.
15              DEC D          ;
20FA            JR NZ,DREGL    ;
D1              POP DE         ; DE umspeichern.
C9              RET
```

4. Schritt

Programm laden, beginnend bei Speicherstelle DISTST. Dieses Programm kann zur Prüfung der Anzeigen des Nanocomputers® benutzt werden. Jede mögliche 7-Bit-Kombination wird in jeder der 10 Anzeigestellungen angezeigt.

Um die Anzeigeneinheit gründlich zu prüfen, gibt es 2^7 verschiedene Möglichkeiten. Sie müßten also 128 mal 10 angezeigte Zeichen beobachten. Dazu benötigen Sie eine relativ lange Zeit; außerdem wird es mit fortschreitender Dauer uninteressant. Deshalb ist das Programm mehr als Übung für den Anzeigenbetrieb des Nanocomputers® gedacht und weniger als ernsthafter Anzeigen-Tester. Führen Sie das Programm aus, was beobachten Sie?

Mehrere Folgen von 10 Anzeigen:

1. Folge: Die LEDs BRK, SP, PC, MEM, I/O, ERR, ARS leuchten gleichzeitig auf. Die LEDs IY, IR, AF, BC, DE, HL, IX leuchten gleichzeitig auf. Wenn Sie von 1-8 von links nach rechts zählen, waren die 7-Segmet-Anzeigen in der nachstehenden Reihenfolge mit einer Ziffer 8 beleuchtet: 5, 6, 7, 8, 1, 2, 3, 4.
2. Folge: Die LEDs BRK, SP, PC, MEM, I/O, ERR leuchten gleichzeitig auf. Die LEDs IR, AF, BC, DE, HL, IX leuchten gleichzeitg auf. Wenn Sie von 1-8 von links nach rechts zählen, waren die 7-Segment-Anzeigen in der nachstehenden Reihenfolge mit einer 0 beleuchtet: 5, 6, 7, 8, 1, 2, 3, 4.
   etc.

In dem vorstehenden Programm werden sowohl die Daten- als auch die Positions-Bytes für jede Schleife zweimal erhöht. Diese doppelte Erhöhung

ist erforderlich, weil D0 weder in der Anzeige noch der gewählten Stellung eine Rolle spielt. Im Falle des Daten-Bytes verhindert die doppelte Erhöhung die Ausgabe eines High-D0-Bits (logisch 1) an das IC HCF4514B. Das ist wichtig, weil sonst die Anzeigen bei jedem nächsten Datenwechsel-Zyklus dunkel bleiben.

5. Schritt.

Wie bereits erwähnt leuchten die 7-Segment-Anzeigen sowie die entsprechenden LEDs nicht alle gleichzeitig, wenn auch dieser Eindruck entsteht. Im Gegenteil, sie werden einzeln und nacheinander angesteuert. Da jedoch die Taktfrequenz sehr hoch ist, entsteht der Eindruck einer flackerfreien Anzeige. Falls Sie die Länge der durch die Subroutine DELAY erzeugten Verzögerung verändern, können Sie die Auswirkung des schnellen Positionswechsels auf der Anzeige sehen. Zu diesem Zwecke ändern Sie das Zeit-Byte F0 in der Subroutine DELAY in 01. Um die Daten auf einen Einzelwert zu halten, ändern Sie die beiden Befehle INC B zu NOPs (hex. 00). Führen Sie das Programm aus. Was beobachten Sie?
Alle LEDs leuchten auf und alle 7-Segment-Anzeigestellungen sind mit der Ziffer 8 beleuchtet. Kein Flackern. Alle Anzeigen scheinen gleichzeitig betrieben zu werden.

6. Schritt

Daten-Byte im Befehl LD BC,0005H durch 54 ersetzen, d.h., der Befehl soll lauten: LD BC,5405H. Führen Sie das Programm noch einmal aus. Was beobachten Sie jetzt?
Folgende LEDs leuchten auf: BRK, MEM, SP, ARS, IR, BC, HL und IY. Das nachstehende Muster erscheint auf allen 7-Segment-Anzeigen:



VERSUCH NR. 2

Er hat die Aufgabe zu verdeutlichen, wie die System-Software des Nanocomputers® eingesetzt wird, um Eingaben vom Tastenfeld anzunehmen und Anzeigen auf der Tastenfeld-Anzeigeeinheit zu erzeugen.

Programm KBTST

Objekt-Code		Quell-Code	Bemerkung
		NAME: KBTST	
CD9DF9	KBTST:	CALL CHECKB	; Auf gedrückte Taste prüfen.
28FB		JR Z,KBTST	; Z-Flag = 1: keine Taste gedrückt. ; Z-Flag = 0: eine oder mehrere Tasten ; gedrückt.
CDDBF8	GETNO:	CALL KBSCAN	; Nachsehen, ob nur eine und welche.

```
38F6                JR C,KBTST      ; C-Flag = 1: mehr als eine Taste gedrückt.
32E20F              LD (DATAL),A    ; C-Flag = 0: eine Taste gedrückt, die
                                    ; entsprechende Ziffer ist in Register A.
                                    ; Hexadezimale Tastenzahl in Anzeige-
                                    ; stellungen anzeigen.
08                  EX AF,AF'       ; Für Aufruf zu CONVDI einrichten.
3EFC                LD A,0FCH       ; Datenziffern anzeigen.
08                  EX AF,AF'
11E50F              LD DE,ADDH
21B90F              LD HL,ADD7-1
CD7CFA              CALL CONVDI     ; Tastenzahl für Anzeige übesetzen.
CD09F9     DSPLAY:  CALL DISPL      ; Tastenzahl anzeigen.
CD9DF9              CALL CHECKB     ; Auf gedrückte Taste prüfen.
28F8                JR Z,DSPLAY     ; weiter anzeigen, falls keine Taste
                                    ; gedrückt
18E1                JR GETNO        ; Falls Taste gedrückt, Zahl feststellen.
```

1. Schritt

Programm laden, beginnend bei Speicherstelle KBTST. In diesem Programm werden mehrere Tastenfeld/Anzeige-Treibroutinen im Betriebssystem des Nanocomputers® benutzt:

CHECKB	zum Aufspüren von gedrückten Tasten
KBSCAN	zur Vergewisserung, ob nur eine Taste gedrückt ist und zur Identifizierung dieser Taste.
CONVDI	zur Übersetzung von Daten aus dem binären oder hexadezimalen Code, in einem für die 7-Segment-Anzeigen geeigneten Code.
DISPL	zum flackerfreien Aufleuchten der LEDs sowie der 7-Segment-Anzeigen

Der logische Ablauf des Programms ist wie folgt:

1. Auf gedrückte Taste prüfen.
2. Feststellen, ob nur eine Taste gedrückt ist, falls ja, identifizieren.
3. Zahl der gedrückten Tasten in den rechtsbündigen zwei 7-Segment-Anzeigeziffern anzeigen.
4. Die vorige Tastenzahl solange anzeigen, bis eine weitere Taste gedrückt wird, dann die neue Zahl anzeigen.

Programm ausführen durch leichtes und schnelles Drücken der G0-Taste. Was beobachten Sie?
Alle Anzeigen erlöschen. Bei etwas festerem Druck auf die G0-Taste wäre in den rechtsbündigen beiden Ziffern der Anzeige 19 erschienen.

2. Schritt

Drücken Sie die Tasten 0, 1, 2, 3, . . . F. Was beobachten Sie?
Auf der Tastenfeld-Anzeige erscheinen entsprechende Ziffern, nämlich 00, 01, 02, 03, . . . 0F. Nach dem Drücken einer Taste ist es möglich, daß die gesamte Anzeige erlischt. Wird dieselbe Taste erneut gedrückt, erscheint die entsprechende hexadezimale Ziffer in der Anzeige sofort. Eine Erklärung für dieses "flackernde" Verhalten ist folgende: Das Programm liest die gedrückte Taste stets zweimal, einmal in der Routine CHECKB und einmal in KBSCAN. Wird die Taste zwischen diesen beiden Ablesungen losgelassen, erkennt das Programm sie nicht als gültige Tasten-

feldeingabe an. Folglich übernimmt die Schleife KBTST die Steuerung, bei der alle Anzeigen dunkel sind, während die Routine auf einen neuen Tastendruck wartet. Eine vollständige Tabelle mit den entsprechenden Hex-Ziffern finden Sie in der Tabelle 5-7.

3. Schritt

Beachten Sie, wie Register A' und die Registerpaare DE und HL vor dem Aufruf zu CONVDI initialisiert werden. Wenn Sie den Inhalt des Registers A' ändern, erhalten Sie gegenüber der Anzeige in Schritt 2 eine andere Ziffernfolge. Ändern Sie z.B. den Befehl LD A, 0FCH in LD A, 0FAH (3EFA) und führen anschließend das Programm aus. Was beobachten Sie? Das rechte 7-Segment-Display zeigt die niederwertigste Ziffer der gedrückten Taste an. Das zweite Display leuchtet nicht auf, während das dritte Display kontinuierlich eine bestimmte Ziffer zeigt. Der Wert der angezeigten Ziffer hängt davon ab, was mit dem Inhalt der Speicherstelle DATAH geschieht.

Tabelle 5-7. Tasten und auf dem Display angezeigte Hex-Ziffern für das Tastenfeld des Nanocomputers®.

Taste	angezeigte Hex-Ziffer	Taste	angezeigte Hex-Ziffer
0	00	→	10
1	01	←	11
2	02	ST	12
3	03	LA	13
4	04	2ND	14
5	05	SS	15
6	06	INC	16
7	07	LD	17
8	08	ARS	18
9	09	GO	19
A	0A	BRK	1A
B	0B	DP	1B
C	0C		1C
D	0D	BREAK	Zufällige Anzeige und Unterbrechung des laufenden Programms.
E	0E		
F	0F	RESET	Rückkehr ins Operationssystem

Fußnote für Experimenter: Um mit der Anzeige etwas zu spielen, können Sie ein Programm laden, daß den Text SGS-NANOROUTINES RELEASE LOADED CIAO schreibt. Es erscheint, wenn Sie das Programm ab F00D in das RAM laden (siehe Anhang B, ab NANOR2).
Das Programm selbst kann von Eintrittspunkt NANOR2 im RAM ausgeführt werden. Sie können die Daten an der Speicherstelle STRING ändern, um irgend etwas anderes zu sagen, was Sie möchten. Die Zeichenreihe kann von beliebiger Länge sein, vorausgesetzt, sie endet mit dem Byte 01H 9 Bytes vor dem Ende.

Kapitel 6

Interrupt-Technik

Stellen Sie sich einmal die nachstehende Ereignisfolge vor:

1. Sie lesen ein Buch.
2. Das Telefon klingelt.
3. Sie markieren die Stelle im Buch und heben den Hörer ab.
4. Sie melden sich am Telefon und deuten so die Bereitschaft an, sich mit dem Anrufer zu unterhalten.
5. Die Unterhaltung beginnt.
6. Es schellt an der Wonhungstür.
7. Sie bitten den Anrufer, einen Augenblick zu warten.
8. Sie öffnen die Tür.
9. Sie unterhalten sich mit der Person an der Tür und erledigen die Angelegenheit mit ihr.
10. Sie kehren zum Telefon zurück und führen das Gespräch zu Ende.
11. Nachdem Sie die markierte Stelle gefunden haben, an der die Unterbrechung begann, lesen Sie weiter.

In der obigen Szene fungieren Sie als Hilfsmittel, das drei Aufgaben zu erledigen hat: Lesen, Telefonieren, und Sprechen mit einer Person an der Haustür. Zu jedem Zeitpunkt entscheiden Sie sich zur Ausführung von nur einer dieser drei Aufgaben. Der Übergang zwischen den Aufgaben wird durch zwei Mechanismen eingeleitet:

a) Unterbrechungen (Interrupts), das Läuten der Türklingel bzw. des Telefons;
b) Aufgabenbewältigung, im Beispiel das Zuendeführen der Gespräche.

Die Übergänge im genannten Beispiel sind geschachtelt. Das heißt: Jede neue Unterbrechung stoppt die Erledigung der momentanen Aufgabe. Sämtliche Hilfsmittel wenden sich der neuen Unterbrechungsaufgabe zu und führen sie zu Ende, sofern keine weitere Unterbrechung eintritt. Anschließend wird die unterbrochene Aufgabe wiederaufgenommen.
Im allgemeinen sind die durch Unterbrechungen verursachten Übergänge von folgenden Ereignissen gekennzeichnet:

a) Die Unterbrechung tritt ein.
b) Die Tätigkeit an der momentanen Aufgabe wird unterbrochen und der augenblickliche Zustand für die spätere Weiterführung festgehalten.
c) Die Unterbrechung wird bestätigt.
d) Es beginnt die Behandlung der von der Unterbrechung eingeleiteten neuen Aufgabe, bis entweder eine neue Unterbrechung eintritt oder die Behandlung beendet ist.

Bei der Erledigung der Aufgaben gilt an den Übergängen folgendes:
a) Die Unterbrechungsaufgabe ist beendet.
b) Der Punkt, an dem eine Unterbrechung die Lösung der Aufgabe stoppt, wird festgehalten.
c) Bei Weiterführung der unterbrochenen Aufgabe stehen die entsprechden Hilfsmittel wieder zur Verfügung.

Zu der eingangs genannten Szene gibt es noch Alternativen. Sie lassen sich beim Lesen des Buches weder durch das Klingeln des Telefons noch durch das Schellen an der Haustür unterbrechen und ignorieren beides. In diesem Fall sind die Unterbrechungen blockiert. Anderseits gibt es Unterbrechungen, die man nicht ignorieren kann, wenn z.B. durch einen elektrischen Kurzschluß das Licht ausgeht. Solche nicht zu ignorierenden Unterbrechungen nennt man *"nichtmaskierbar"*. Man muß diese Unterbrechungen behandeln (Kurzschluß beheben), bevor man sich wieder der ursprünglichen Aufgabe zuwendet. Nichtmaskierbare Unterbrechungen haben eine höhere Priorität gegenüber den Unterbrechungen, die man ignorieren kann *(maskierbare Unterbrechungen)*.
Innerhalb der maskierbaren Unterbrechungen gibt es Aufgaben mit höherer Priorität, die keine Unterbrechung dulden und solche mit geringerer Priorität. So kann es z.B. unerwünscht sein, ein Ferngespräch zu unterbrechen, weil es an der Tür gekopft hat. Die andere Extrem: Wenn Sie sich durch die geringste Zerstreuung unterbrechen ließen, hätten Sie die ganze Zeit nichts anderes zu tun, als Unterbrechungen zu bewältigen und kämen nie zum Lesen. In diesem Falle könnte man sagen, Sie sind *unterbrechungsorientiert*.
Der eigentliche Sachverhalt und wesentliche Kernpunkt bei der Interrupt-Technik ist die *Systemelement-Verteilung*. Sie sind das Systemelement, das mehreren unterschiedlichen Aufgaben zugeteilt ist. Es gibt einen Wettstreit zwischen den Aufgaben für die begrenzten, zur Verfügung stehenden Systemelemente. Es ist wünschenswert die Probleme in einer Weise zu lösen, welche die Leistungsfähigkeit des Systems optimal nutzt.
Die bisherigen Aussagen treffen ebenso gut auf das Problem zu, CPU-Systemelemente auf mehrere unterschiedliche Aufgaben aufzuteilen. Für einen Großcomputer sind z.B. die unterschiedlichen Aufgaben Einzelbenutzer an entfernten Datenstationen, die versuchen, Aufträge zu erteilen und auszuführen. Für einen Mikrocomputer können die Aufgaben im Datentransfer zwischen mehreren verschiedenen I/O-Geräten bestehen, die eine Technik benutzen, die sich interruptbetriebene Ein-/Ausgabe nennt.
In diesem Kapitel werden zunächst alternative Techniken für CPU-Systemelement-Verteilung erörtert. Die besondere Betonung liegt dabei auf Interrupt-Technik, weil dies die häufigste Technik für Mikrocomputer-Anwendungen ist. So ist es selbstverständlich, daß das Kapitel anhand von Versuchen näher auf die Interrupt-Technik mit der Z-80-CPU eingeht.
Am Ende dieses Kapitels werden Sie zu folgendem in der Lage sein:
- die Begriffe *Interrupt* (Unterbrechung) *Abrufen* und *direkter Speicherzugriff* als CPU-Systemelement-Verteilungstechniken zu definieren;
- die relativen Verdienste der Abruftechnik und Interrupt-Technik im Hinblick auf die Benutzung von CPU-Systemelementen zu erörtern;
- die Begriffe *maskierbare* und *nichtmaskierbare Interrupts* zu definieren;
- die drei bekannten Methoden der Interrupt-Technik zu definieren:

einzeilige Unterbrechung, mehrstufige Unterbrechung und *gerichtete Unterbrechung;*

- die Begriffe Kontext-Schaltung und ablaufvarianter Code zu definieren und
- die erforderlichen Schaltungen für die Durchführung der drei Unterbrechungsmethoden der Z-80-CPU zu verstehen und herzustellen.

ARTEN VON MIKROCOMPUTER-OPERATIONEN

In diesem Buch und dem Buch 1 für den Z-80-Betrieb ist die Software durch die Eigenschaft charakterisiert, daß ein gestartetes Programm bis zum Ende durch läuft. Das ist die einfachste Computer-Betriebsart. Hierbei sind alle Systemelemente des Computers darauf gerichtet, die laufende Aufgabe zu erfüllen und können nicht umverteilt werden, bis die Aufgabe abgeschlossen ist und die Steuerung freigibt. Trotzdem haben Sie bereits ein Beispiel der Interrupt-Technik kennengelernt, ohne sich weiter damit zu befassen. Die SS-Taste nutzt die Z-80 Interruptmöglichkeiten, um den Transfer der Steuerung zwischen dem Anwenderprogramm und dem Operationssystem des Nanocomputers® zu bewerkstelligen. Da die angewendeten Techniken auch von den Eigenschaften des PIO-ICs abhängig sind, wird an dieser Stelle auf eine weitere Beschreibung verzichtet.
Man unterscheidet grundsätzlich zwei Arten von Z-80 Interrupt-Techniken; beide sind in diesem Kapitel ausführlich behandelt.

SOFTWAREGESTEUERTE TECHNIK – Es handelt sich dabei um eine Technik, bei der alle Aufgabenübergänge ausschließlich durch Softwaresteuerung stattfinden. Beispiele für softwaregesteuerte Übertragungen sind Subroutine-Abrufe sowie Sprünge und Rücksprünge. Eine gängige Art zur Durchführung von softwaregesteuerten Übertragungen stellt die Abruftechnik dar.

HARDWARE- UND SOFTWAREGESTEUERTE TECHNIK – Eine Technik, bei der die Aufgabenübertragung entweder durch Software oder durch Hardware gesteuert wird. Beispiele für hardwaregesteuerte Übertragungen sind Unterbrechungen (Interrupts) und Datenübertragungen durch direkten Speicherzugriff (DMA = direct memory access). Die beiden externen Z-80-ICs – PIO und DMA – sind speziell für Interrupt- und DMA-Anwendungen konzipiert.
Nachstehend eine Definition einiger zuvor benutzter Begriffe:
Abrufen – Das Abrufen bezieht sich auf eine Software-Technik, die feststellt, ob ein externes Gerät bedient werden muß. Das externe Gerät ist über eine Verbindungsleitung mit der CPU gekoppelt. Die regelmäßige Überprüfung gilt den Signalen, die auf eine Bedienung des Gerätes hinweisen.
Interrupt – Ein Signal, das einer CPU eingegeben wird, um anzuzeigen, daß ein externes Gerät Bedienung benötigt. Bei der Ausführung eines Befehls tastet die CPU immer ihren Interrupt-Eingang ab. Bei einem aktivierten Eingang beginnt die Interrupt-Behandlung, sobald der momentan ausgeführte Befehl beendet ist.

Direkter Speicherzugriff (DMA) – Diese Technik überträgt große Datenblöcke zwischen der CPU und einem externen Gerät. Nach Einleitung durch die CPU wird die Übertragung vollständig von einer speziellen externen Schaltung abgewickelt. Dadurch sind keine Systemelemente der CPU für die Übertragung erforderlich. Da aber Adreß-, Daten- und Steuer-BUS an dieser Datenübertragung beteiligt sind, muß die CPU oft mit dem DMA-Gerät zusammenarbeiten.

Nachstehend ein Beispiel zur Verdeutlichung der Begriffe Abrufen, Interrupt und direkter Speicherzugriff:

Wenn Sie Telefongespräche nach der Abruftechnik annehmen, müßten Sie den Hörer von Zeit zu Zeit abnehmen und fragen "Ist dort jemand?"

Wenn Sie Telefongespräche nach der Interrupt-Technik annehmen, warten Sie bis das Telefon klingelt und nehmen dann den Hörer ab.

Nehmen Sie Telefongespräche nach der direkten Speicherzugriffstechnik an, speichern Sie die Gespräche auf ein Tonband und hören Sie nach Bedarf ab. Das schließt natürlich einen intensiven Informationsaustausch zwischen Ihnen und dem Anrufer aus.

Die vorstehenden Techniken stellen drei verschiedene Wege zur Verteilung der CPU-Systemelemente auf potentielle Verbraucher dar. Der Abruf ist der am meisten zentralisierte Weg, da die CPU jeden Benutzer einzeln fragt, ob eine Bedienung erforderlich ist. Wünscht ein Benutzer die Bedienung, so erfolgt diese, bevor der nächste Benutzer danach gefragt wird. Es gibt also kein Drängeln zwischen den Benutzern. Sie werden in der richtigen Reihenfolge bedient, einer nach dem anderen.

Im Gegensatz dazu erfordert die Interrupt-Technik eine aktivere Mitwirkung des externen Geräts bei der Abforderung von Systemelementen. Damit eine Bedienung möglich ist, muß eine Programmunterbrechung (Interrupt) erfolgen. Bei einem nichtvorrangigen Interrupt erkennt die CPU jede Unterbrechung während sie stattfindet und beginnt sofort mit der Bearbeitung. Die letzte Bearbeitung wird fortgesetzt und bis zu Ende geführt oder von einem neuen Interrupt unterbrochen. Was geschieht, wenn zwei Geräte eine Bearbeitung gleichzeitig anfordern? In fast allen Arbeitssystemen hat man allen potentiellen Interrupt-Geräten Prioritäten eingeräumt. Das Gerät mit der höchsten Priorität führt seine Aufgabe bis zu Ende aus, es sei denn, es folgt ein Interrupt mit noch höherer Priorität. Der Weg zur Zuweisung von Systemelementen ist weniger zentralisiert als beim Abrufen, da die externen Geräte die Initiatoren der Kommunikation sind. Der DMA-Weg addressiert das Problem der Systemelement-Zuweisung durch Hinzufügen eines wirkungsvollen Satzes neuer Systemelemente, nämlich der Großspeichersteuerung. Bei DMA handelt es sich nicht um eine Allzwecktechnik wie das Abrufen und die Interrupt-Technik. Der direkte Speicherzugriff hat die Aufgabe, große Datenblöcke zwischen dem CPU-Speicher und den externen Geräten mit Hilfe von Systemelementen zu übertragen. Somit verkörpert der direkte Speicherzugriff (DMA) eine Strategie, welche die vorhandenen Systemelemente durch neue, hochspezialisierte Fähigkeiten bereichert. DMA stellt also für alle Verbraucher von Systemelementen eine Teilmenge dar.

Aus dem Telefonbeispiel lassen sich sofort mehrere Vorteile bei der Interrupt-Technik gegenüber der Abruftechnik erkennen. Der Hauptnachteil der Abruftechnik ist die Benutzung von Systemelementen, auch

wenn keine Bearbeitung erforderlich ist. Ein weiterer Nachteil, die Antwortzeit kann relativ lang sein. Wenn Sie in dem Telefonbeispiel die Leitung nur einmal in der Stunde abfragen, müßte der Anrufer möglicherweise 59 Minuten warten, bis er mit Ihnen sprechen kann.
Die Interrupt-Technik hat gegenüber der Abruftechnik mehrere wichtige Vorteile. Erstens ist die Verteilung der Systemelemente streng auf Bedarf abgestellt. Es werden keine Systemelemente für Geräte verschwendet, die keine Bearbeitung benötigen. Eine Bearbeitung kann beginnen, sofort nachdem ihre Notwendigkeit erkannt ist. Die Zeit bis zur Beantwortung ist daher relativ kurz. Der Nachteil der Interrupt-Technik: sie kann den erforderlichen Software-Aufwand beträchtlich erhöhen. Bei den Interrupts handelt es sich um Zufallsereignisse, die ein Programm bei jeder Ausführungsstufe unterbrechen können. So ist manchmal zusätzliche Software-Logik erforderlich, damit das Programm die Ausführung wieder aufnehmen kann, nachdem die Unterbrechung abgearbeitet ist. Darüber hinaus kann der Zufallscharakter die Reproduktion, Lokalisierung und Beseitigung von intermittierenden Software-Programmfehlern sehr erschweren. Ein weiterer Nachteil: die Interrupt-Technik erfordert oft teure Hardware zur Prioritätseinstufung und Identifikation von Interrupt-Geräten. Viele dieser Situationen machen die späteren Versuche deutlich.
Die direkte Speicherzugriffstechnik (DMA) für große Blöcke von I/O-Übertragungen zwischen der CPU und den externen Geräten basiert hauptsächlich auf der zusätzlichen Einfügung eines weiteren Systemelements im Gegensatz zur Verteilung der CPU-Systemelemente. Weil sich die CPU und die Großspeichersteuerung den Adreß-, Daten- und Steuer-BUS teilen, macht die DMA beinahe eine weitere CPU erforderlich, die speziell I/O-Übertragungen realisiert. Der Hauptvorteil des DMA ist seine geringe Benutzung der CPU-Systemelemente. Doch sind Kosten und die Komplexität bei der Auswahl der Technik oft die wichtigsten Gesichtspunkte.

INTERRUPT-GRUNDARTEN

Alle Interrupts können in *maskierbare* und *nichtmaskierbare* eingeteilt werden:
Maskierbare Unterbrechung – Ein Interrupt ist maskierbar, wenn mit Hilfe einer Anweisung die softwaregesteuerte CPU den Interrupt ignorieren kann. Die Z-80-CPU hat eine maskierbare Interrupt-Leitung, die mit $\overline{INT}$ bezeichnet ist. Der Befehl für die Z-80-CPU alle maskierbaren Interrupts zu ignorieren lautet DI (Disable Interrupts - Unterbrechungen blockieren). Der Befehl zur Duchführung der maskierbaren Interrupts lautet EI (Enable Interrupt - Unterbrechungen freigeben). Auf diese Befehle wird später noch ausführlicher eingegangen.
Nichtmaskierbare Unterbrechung – Ein Interrupt ist nichtmaskierbar, wenn die CPU ihn unter keinen Umständen ignorieren darf. Die Z-80–CPU hat eine nichtmaskierbare Interrupt-Leitung, die mit $\overline{NMI}$ bezeichnet ist.
Für die meisten Geräte, die auf Unterbrechungsbasis von der CPU bedient werden, benutzt man nichtmaskierbare Interrupts. Dadurch kann man über

die Software die Interrupt-Bearbeitung durch die CPU steuern. Die erste Aufgabe einer Interrupt-Bearbeitung besteht darin, den Zustand der CPU zwischenzuspeichern. Das ist notwendig, um die unterbrochene Aufgabe nach der Interrupt-Bearbeitung zu beendigen. Außerdem ist es wünschenswert, maskierbare Interrupts bei der Durchführung der Zwischenspeicherung zu blockieren.

Eine gängige Anwendung für nichtmaskierbare Interrupts ist das Aufspüren von Stromausfällen. Der Mikroprozessor Z-80 wird mit +5 V Gleichspannung versorgt. Das Netzgerät bezieht die notwendige Versorgungsspannung aus dem 220 V-Wechselstromnetz. Eine Stromabfall-Aufspürschaltung löst einen nichtmaskierbaren Interrupt aus, sobald die Wechselspannung auf unter 80% des normalen Wertes abfällt. Bevor der Spannungsverlust für die Aufspürschaltung selbst relevant wird, warnt sie die CPU. Ehe die Gleichspannung unter +5 V abfällt, tritt eine Verzögerung von einigen Millisekunden ein. In dieser Zeit kann die CPU ihren Zustand zwischenspeichern. Dies ist bei großen Prozeßsteuerungsanlagen von immenser Wichtigkeit.

Die maskierbaren und nichtmaskierbaren Interrupts teilen sich noch in verschiedene Kategorien. Sie unterscheiden sich in der Interrupt-Behandlung, wie z.B. die CPU und das Interrupt-Gerät eine so wichtige Information weiterleiten:

Vom Gerät zur CPU:

a) Bearbeitungsanforderung
b) Identifizierung, damit die CPU weiß, welche Arbeit sie ausführen soll.

Von der CPU zum Gerät:

a) Bestätigung der Unterbrechung,
b) andere Daten, soweit notwendig.

Es gibt drei Arten von Interrupts:

single-line interrupt (Einstufige Unterbrechung) — Die CPU-Eingabe zeigt an, wann ein Interrupt gewünscht wird. Ist nur ein Interrupt-Gerät im System vorhanden, ist dies der einfachste und direkteste Weg zur Durchführung der Interrupt-Bearbeitung. An dieser Leitung müssen mehrerer Interrupt-Geräte angeschlossen sein, damit jeder Interrupt direkt zur CPU gelangt. Hat die CPU den Interrupt registriert, muß sie alle Geräte abtasten, um den Auslöser des Interrupts festzustellen.

multilevel interrupt (Mehrstufige Unterbrechung) — Dabei handelt es sich um mehrere CPU-Eingaben, die eine Unterbrechung anzeigen können. In diesem Fall ist jedes Gerät an eine separate Interrupt-Leitung angeschlossen. Die CPU braucht die einzelnen Geräte nicht mehr nach dem Verursacher des Interrupts abzutasten.

vectored interrupt (Angezeigte Unterbrechung) — Die Unterbrechung besteht nicht nur aus dem Anforderungsignal. Für die CPU ist gleichzeitig ein Hinweis vorhanden, welche Interrupt-Bearbeitungsroutine für sie in Frage kommt.

Die drei Interrupt-Arten sind schematisch in Bild 6-1 dargestellt. Die einstufige Unterbrechung stellt eine weit verbreitete Interrupt-Art für

Mikroprozessoren dar, weil sie einfach und billig zu realisieren ist. Die Anzahl der angeschlossenen Geräte ist hierbei unbegrenzt. In Bild 6-1 sind drei angeschlossene Geräte symbolisiert. Es ist jedoch klar, daß mit steigender Zahl der angeschlossenen Geräte auch die Interrupt-Antwortzeit zunimmt.

Die mehrstufige Unterbrechung ist eine gängige Technik, wenn der Mikroprozessor über genügend Anschlußmöglichkeiten verfügt. In der Regel verfügen die gängigen Mikroprozessor-ICs hierfür über nicht mehr als vier Anschlüsse.

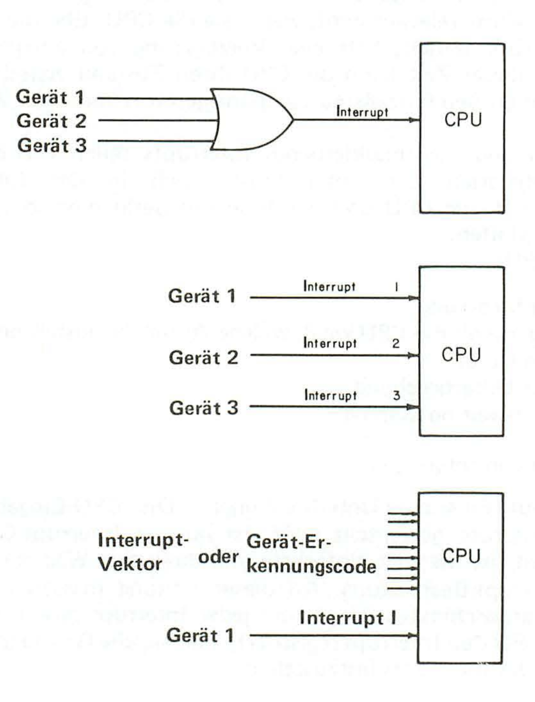


Bild 6-1. Die Skizzen verdeutlichen die drei unterschiedlichen Interrupt-Arten.

Die angezeigte Unterbrechung ermöglicht dem Gerät eine direkte Verzweigung zur Interrupt-Bearbeitungsroutine. Bei einem 8-Bit Mikroprozessor geht zuerst der Interrupt-Impuls zur CPU, gefolgt von einem Gerätecode. In der Praxis ist der 8-Bit-Code entweder eine Adresse, ein Adressenhinweis oder das erste Byte eines Befehls.

Die Z-80-CPU bevorzugt einstufige und angezeigte Unterbrechungen. Die angezeigte Unterbrechungsmethode akzeptiert einen 8-Bit-Erkennungscode, bei dem es sich um eine Hinweisadresse für den Anfang eines Interrupt-Programms oder um das erste Byte eines Z-80-Befehls handelt.

WELCHES SIND DIE CHARAKTERISTIKEN DER INTERRUPT-MÖGLICHKEITEN BEI DER Z-80-CPU?

Der folgende Abschnitt beantwortet eine Reihe allgemeiner Fragen, die vor die Charakterisierung der Interrupt-Möglichkeiten bei der Z-80- und anderen CPUs relevant sind.

1. Wie erkennt die CPU eine Interrupt-Anforderung? Die Z-80-CPU hat zwei Anschlußstifte, die mit $\overline{INT}$ und $\overline{NMI}$ bezeichnet sind. Sie werden von externen Geräten zur Anforderung von Interrupts benutzt. Mit der ansteigenden Flanke des letzten Taktes im T-Zyklus eines Befehls tastet dieZ-80-CPU beide Anschlußstifte ab. Ist einer aktiv (im Zustand logisch 0), wird eine Unterbrechung angefordert. Daher beträgt die maximale Zeitspanne zwischen einer Unterbrechungs-Anforderung und ihrer Erkennung durch die CPU einen Befehlszyklus. Die Befehlszuklen dauern zwischen 4 und 23 T-Zyklen, so daß die Erkennungsverzögerung bei einem Interrupt zwischen 1,6 bis 9,2 Mikrosekunden beträgt. Eine interessante Frage, die bei der Erörterung der Unterbrechungserkennung durch die CPU auftaucht: Was geschieht mit den "langen" Befehlen LDIR, LDDR, CPIR, CPDR, OTIR, OTDR, INIR und INDR? Eine irrige Meinung ist, daß bei diesen Befehlen die Signale $\overline{INT}$ und $\overline{NMI}$ nicht abgetastet werden, kurz bevor das Registerpaar BC auf 0000 geht. Im Gegenteil, die Z-80-CPU behandelt die Ausführung all dieser Befehle wie eine *Reihe von bestimmten Befehlszyklen.* Ein Befehlszyklus entspricht der Bewegung eines einzelnen Bytes. Daher kann eine Unterbrechung mitten in der Ausführung eines dieser Befehle eintreten. Wenn die CPU eine Interrupt-Bearbeitung zu Ende führt, wird die Ausführung desselben "langen" Befehls vom Unterbrechungspunkt aus fortgesetzt, falls der Zustand aller beteiligten Register zwischengespeichert ist.

2. Werden nichtmaskierbare Unterbrechungen unterstützt? Falls ja, wieviele? Die Z-80-CPU unterstützt eine nichtmaskierbare Unterbrechung. Die nichtmaskierbare Unterbrechung gelangt über den Eingangspin 17 (mit $\overline{NMI}$ bezeichnet) zur Z-80-CPU.
   Beim nichtmaskierbaren Interrupt handelt es sich um eine einstufige Unterbrechung. Ist der Pin $\overline{NMI}$ aktiv (Zustand logisch 0), erkennt die Z-80—CPU die Anforderung eines nichtmaskierbaren Interrupts, bestätigt den Empfang und leitet den Interrupt zur Speicherstelle 0066. Hat die CPU den zum Unterbrechungszeitpunkt ausgeführten Befehl beendet, fügt sie den Befehl CALL 0066H logisch in das laufende Programm ein. Als Ergebnis überträgt die CPU die Steuerung vorübergehend an eine Subroutine, welche die nichtmaskierbare Unterbrechung

bearbeitet. Ist das Interrupt-Programm beendet, folgt ein Rückspungbefehl zum unterbrochenen Programm.
Einzelheiten dieses Vorgangs werden bei den Versuchen am Ende dieses Kapitels erörtert. Es ist wichtig zu wissen, wie man die Subroutine für die Steuerung von Übertragungen zwischen dem unterbrochenen Programmunterbrechung benutzt.

3. Werden maskierbare Unterbrechungen unterstützt? Falls ja, wieviele? Die Z-80 unterstützt zwei Arten von maskierbaren Unterbrechungen, und zwar einstufige und angezeigte Unterbrechungen. Die angezeigten Unterbrechungen teilen sich in zwei Arten, in die *Interrupt-Methode 0* und *Interrupt-Methode 2.* Die Methode 0 erwartet von dem Interrupt-Gerät einen Befehl, während die Methode 2 eine Adresse erwartet. Die Interrupt-Bearbeitung nach Methode 1 benutzt die einstufige Unterbrechung, bei der ein Sprung zur Speicherstelle 0038 automatisch erfolgt. Alle drei nichtmaskierbare Unterbrechungsarten werden über Pin 16 ($\overline{INT}$) der Z-80-CPU realisiert. Die übliche Unterbrechungsmethode ist softwaregesteuert. Das heißt, daß je nach Programmierung die CPU ein low-aktives Signal am $\overline{INT}$-Eingang als Interrupt-Methode 0, 1 oder 2 interpretiert. Drei Befehle bestimmen die Interrupt-Methode: IM0 = ED46H, IM1 = ED56H und IM2 = ED5EH. Die Tabelle 6-1 zeigt eine Zusammenfassung der maskierbaren und nichtmaskierbaren Unterbrechungen der Z-80. Auch hierfür finden Sie die Einzelheiten bei den Versuchen am Ende dieses Kapitels.
4. Woher weiß die CPU, welches Gerät den Interrupt wünscht? Für einstufige Unterbrechungen können ein oder mehrere Geräte am $\overline{NMI}$- oder $\overline{INT}$-Pin angeschlossen sein. Ist nur ein Gerät angeschlossen, gibt es keine Identifikationsprobleme. Können mehrere Geräte den Interrupt-Eingang in den Zustand logisch 0 versetzen, sind zusätzliche Schaltungen erforderlich. Die CPU benötigt dann mehr an Information als nur das Signal am entsprechenden Interrupt-Eingang. Im allgemeinen besteht diese zusätzliche Schaltung aus einem Eingangsgatter, das ein Bit mit jedem Interrupt-Gerät verbindet. Das Interrupt-Gerät könnte sein Bit (EXTERNAL FLAG = EXTERNES MERKBIT genannt) in den Zustand logisch 1 versetzen, während die Bits der nichtunterbrechenden Geräte im Zustand logisch 0 bleiben. Daüber hinaus kann die CPU auch mit Software bei mehreren Interrupt-Geräten das richtige auswählen. Die Software setzt dabei Prioritäten. Sie ist noch intelligenter, wenn sie die nach dem letzten Abruf verstrichene Zeit berücksichtigt, die für die Interrupt-Bearbeitung erforderliche Zeit sowie die System-Prioritäten dynamisch ändert.
Für die hardwaremäßige Prioritätsauswahl existieren auch integrierte Schaltungen, deren Vorteil gegenüber der Software-Lösung die höhere Geschwindigkeit ist. Demgegenüber hat die Software den Vorteil der höheren Flexibilität. Für angezeigte Unterbrechungen dient der 8-Bit-Befehl (Methode 0) oder die Adresse (Methode 2) als Geräte-Bezeichner. Im wesentlichen teilt der Geräte-Bezeichner der CPU mit, welche Programmunterbrechung auszuführen ist. Darum ist der Befehl nach der Interrupt-Methode 0 in der Regel ein Verzweigungsbefehl, (z.B. RST oder CALL), während die nach der Interrupt-Methode 2 erteilte Adresse auf den ersten Befehl der Programmunterbrechung hinweist.

Tabelle 6-1. Zusammenfassung der maskierbaren und nichtmaskierbaren Unterbrechungen bei der Z-80-CPU.

Methode	Aktion	Zurück
NMI	CALL zur Speicherstelle 0066	RETN
IMO	Daten-BUS enthält nächsten Befehl	RETI
IM1	CALL zur Speicherstelle 0038	RETI
IM2	Der Daten-BUS führt die niedrigwertigen 8 Bits der Vektor-Adresse. Register I enthält die 8 höchstwertigen Bits der Vektor-Adresse. I-Register D0 . . . D7 Vektor-Adresse LO HI CALL zu dieser Adresse	RETI

5. Wie löst die CPU das Problem bei zwei gleichzeitig stattfindenden Interrupts? Für den Z-80 hat die $\overline{NMI}$-Leitung gegenüber der $\overline{INT}$-Leitung Vorrang. Daher kann eine nichtmaskierbare Unterbrechung die Bearbeitung einer maskierbaren Unterbrechung stoppen. (siehe Versuch Nr. 4). Wie bereits erwähnt, müssen Rangeleien zwischen zwei oder mehreren nichtmaskierbaren oder zwischen zwei oder mehreren maskierbaren Unterbrechungen über Hardware, Software oder eine Kombination von beiden gelöst werden.

6. Wie wird die Annahme eines Interrupts bestätigt? Die Z-80-CPU bestätigt die Annahme von Unterbrechungen, indem sie einen speziellen M1-Zyklus – den *Interrupt-Bestätigungszyklus* – ausführt. Bei diesem M1-Zyklus werden die Signale $\overline{IORQ}$ (anstelle des normalen Signals $\overline{MREQ}$) und $\overline{M1}$ aktiviert, um das low-aktive Interrupt-Bestätigungssignal $\overline{INTA}$ zu bilden.

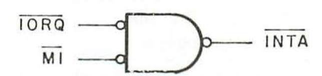


Bild 6-2. Schaltung für das Low-Interrupt-Bestätigungssignal mit einem ODER-Gatter.

Auf die Interrupt-Anforderungs-/Bestätigungszyklen der Z-80-CPU gehen die Versuche näher ein.

7. Wie werden maskierbare Unterbrechungen freigegeben und blockiert? Maskierbare Unterbrechungen werden sowohl durch die Z-80-CPU als auch von der Software-Steuerung automatisch freigegeben und blokkiert. Zwei interne CPU-Flipflops informieren darüber, ob maskierbare Unterbrechungen freigegeben oder blockiert sind. Es handelt sich dabei um die Flipflops IFF1 und IFF2. IFF ist die Abkürzung für *Interrupt Flipflop.*
IFF1 steuert die Aktivierung der maskierbaren Unterbrechungen, während IFF2 manchmal als Speicher für IFF1 dient. Die CPU steuert IFF1 und IFF2 unter folgenden Bedingungen:

a) Ist die CPU zurückgesetzt, gilt das auch für IFF1 und IFF2, damit maskierbare Unterbrechungen blockiert werden.
b) Hat die CPU einen Interrupt angenommen, bleiben beide Flipflops zurückgesetzt; dadurch sind maskierbare Unterbrechungen blockiert.
Für die Freigabe und Blockierung von maskierbaren Unterbrechungen empfängt die CPU zwei Befehle: DI (F3H) für die Blockade und für die Freigabe EI (FBH). Der EI-Befehl setzt beide Flipflops, während der DI-Befehl beide zurücksetzt. Eine wichtige Eigenschaft des EI-Befehls ist die einmalige Befehlsverzögerung, bevor die Z-80-CPU den Interrupt-Befehl annimmt. Bei der Befehlsfolge

```
EI
RETI
```

wird die maskierbare Unterbrechung erst angenommen, nachdem der RETI-Befehl bereits ausgeführt ist.
(RETI ist der Befehl "von der Unterbrechung zurückkehren", den man anstelle des populäreren RET-Befehls bei Programmunterbrechungen benutzt. An späterer Stelle wird hierauf noch eingegangen.) Die einmalige Befehlsverzögerung zur Freigabe von Unterbrechungen ist aus

folgendem Grund wichtig: Programmunterbrechungen enden häufig mit der obigen Befehlsfolge. Durch die Befehlsverzögerung wird die Rückkehr von der Unterbrechung vor der Annahme einer weiteren Unterbrechung vollzogen. Dadurch erübrigt sich ein unnötiger Aufbau des Stapelspeichers durch Unterbrechungen, die während der Ausführung des EI-Befehls eintreten. Der DI-Befehl wirkt augenblicklich!

In der Tabelle 6-2 ist eine Zusammenfassung der Z-80-Befehlsgruppe für die CPU-Steuerung zusammengestellt.

Tabelle 6-2. CPU-Steuergruppe

Befehl	Code	
"NOP"		
"HALT"		
DISABLE INT "(D1)"		
ENABLE INT "(EI)"		
SET INT MODE 0 'IM0'	ED 46	8080A - Modus
SET INT MODE 1 'IM1'	ED 56	Sprung zur Speicherstelle 0038H
SET INT MODE 2 'IM2'	ED 5E	Indirekter Sprung. Benutzt Register I und 8 Bits vom Interrupt-Gerät als Zeiger.

EINFÜHRUNG IN DIE VERSUCHE

Während das Grundkonzept der Interrupt-Bearbeitung relativ einfach ist, läßt es sich oft nicht leicht verwirklichen. Die vorhergehenden Abschnitte haben nur Konzepte dargelegt und auf Einzelheiten verzichtet. Die Versuche in diesem Kapitel haben die Aufgabe, die Eigenschaften der Interrupt-Bearbeitung für die Z-80-CPU im Einzelnen aufzuzeigen. Dabei kommen möglichst viele Details zur Sprache. Zu diesem Zwecke wird eine ganze Reihe von Software-Routinen herangezogen. Sie zeigen Ihnen was genau in einer CPU und in einem Speicher vorgeht, wenn sie die Unterbrechung annehmen, bestätigen und bearbeiten. Software, die eine dynamisch wechselnde Umgebung kontrolliert und anzeigt, führt eine ECHTZEIT-BEARBEITUNG durch. Wenn sich die Umgebung nicht schneller ändert, als die Software diese Änderungen tasten und melden kann, zeigt die Echtzeitstatus-Meldung stets den momentanen Zustand der Umgebung an. Ist andererseits die Änderung schneller, als die Software sie registriert, wird die Statusmeldung ungenau. Die verstrichene Zeit zwischen der Umgebungsänderung und Ausführung der erforderlichen Programmschritte ist wichtig, weil sich die Umgebung in der Zeit bereits

wieder verändern kann. Die Statusanzeige ist *keine* genaue Wiedergabe des augenblicklichen Umgebungszustandes. Selbst bei sehr schnellen Computern besteht bei der Echtzeit-Statuskontrolle zwischen Beobachtung und Anzeige immer eine Verzögerung. Bei einigen Systemen beträgt die Verzögerung sogar Stunden, was jedoch immer noch als akzeptable Leistung angesehen wird. Andere Systeme benötigen wiederum nur einige Mikrosekunden, was immer noch nicht schnell genug ist. Die in den folgenden Versuchen benutzte Echtzeit-Software versucht, den momentanen Zustand der CPU festzuhalten, während sich die Umgebung verändert. In den meisten Fällen beträgt die Zeit von der Beobachtung bis zur Meldung Mikrosekunden und ist vernachlässigbar. Treten aufgrund der Zeitverzögerungen nennenswerte Verzerrungen auf, soll darauf hingewiesen werden. Nachstehend finden Sie eine Kurzbeschreibung der folgenden Versuche:

Versuch Nr.	Bemerkung
1	Demonstriert die einstufige Z-80-Unterbrechung, d.h. die Bearbeitung nach Methode 1;
2	demonstriert die Bearbeitung der Z-80-Unterbrechung nach Methode 0;
3	demonstriert die Bearbeitung der Z-80-Unterbrechung nach Methode 2;
4	demonstriert die nichtmaskierbare Unterbrechung der Z-80;
5	behandelt den Begriff KONTEXT-SCHALTUNG und demonstriert zwei Techniken bei der Z-80-Interrupt-Bearbeitung;
6	behandelt die beiden Begriffe GESCHACHTELTE UNTERBRECHUNGEN und ABLAUFVARIANTE. Die Konzipierung eines ablaufvarianten Programms demonstriert eine Verteil-Technik für Programmunterbrechungen bei Mehrfach-Interrupt-Geräten.

VERSUCH NR. 1

Der Versuch macht Sie mit vielen Einzelheiten der nichtmaskierbaren Interrupt-Bearbeitung des Z-80 bekannt. Außerdem zeigt er die Interrupt-Bearbeitung nach Methode 1.

Programme: INIT1, MAIN, und SERV1

```
Objekt-Code            Quell-Code                  Bemerkung
-------------------------------------------------------------------------------
                       NAME INIT1
3EC3        INIT1:     LD A,0C3H              ; erstes Byte ist Sprung
323800                 LD (0038H),A
FD216E02               LD IY,SERV1            ; Adresse der Service-
FD223900               LD (0039H), IY         ; Routine Nr. 1.
ED56                   IM1                    ; Interrupt-Methode 1.
08                     EX AF,AF'              ; Format für Leerraum.
3E40                   LD A,40H               ; für CONVDI setzen.
08                     EX AF,AF'
C3C302                 JP MAIN                ; Zur Main-Routine springen.
                       NAME MAIN
FB          MAIN:      EI                     ; Interrupt freigeben.
DD21000C               LD IX,DSTACK           ; Boden des Daten-Stapelspeichers
DD3600FF               LD (IX+00H),0FFH       ; Zeitgeber für Anzeige.
21E50F                 LD HL,ADDH             ; Hinweisadresse zum Puffer setzen.
ED57                   LD A,I                 ;Wert von IFF2 suchen.
EAD802                 JP PE,HIGH
3600        LOW:       LD (HL),00H            ;Wert = 0.
1802                   JR NEXT
3610        HIGH:      LD (HL),10H            ;Wert = 1.
2B          NEXT:      DEC HL                 ; Puffer-Zeiger schieben.
35                     DEC (HL)               ; COUNT erniedrigen.
ED73E20F               LD (DATAL),SP          ; SP zum Puffer übertragen.
21B90F                 LD HL,LEDL             ; Für CONVDI einrichten.
11E50F                 LD DE,ADDH             ; Für CONVDI einrichten.
00          DISAB:     NOP                    ; Keine Operation.
CD7CFA                 CALL CONVDI
CD09F9      DLOOP:     CALL DISPL
DD3500                 DEC (IX+00H)           ; Zeitgeber für Anzeige.
20F8                   JR NZ,DLOOP
C3C302                 JP MAIN                ; Rücksprung auf Routine MAIN.
                       NAME SERV1
C5          SERV1:     PUSH BC                ; CPU-Register zwischen-
D5                     PUSH DE                ; speichern.
E5                     PUSH HL
F5                     PUSH AF
DDE5                   PUSH IX
FDE5                   PUSH IY
DD23        DS1:       INC IX                 ; Daten-Stapelspeicher
DD23                   INC IX                 ; aktualisieren.
DD23                   INC IX
00                     NOP                    ; keine Operation.
DD3600FF               LD (IX+00H),0FFH       ; DLOOP1-Zeit setzen.
DD36010A               LD (IX+01H),00AH       ; CLOOP1-Zeit setzen.
DD360202    CLOOP1:    LD (IX+02H),02H        ; DLOOP1-Zeit setzen.
21E50F                 LD HL,ADDH             ; Hinweis auf Anzeige-Puffer.
ED57                   LD A,I                 ;Wert von IFF2 suchen.
EA9502                 JP PE,HIGH1
3600        LOW1:      LD (HL),00H            ;Wert = 0.
1802                   JR NEXT1
```

```
3610       HIGH1:  LD (HL),10H      ; Wert = 1.
2B         NEXT1:  DEC HL           ; Puffer-Zeiger schieben.
34                 INC (HL)         ; ADDL erhöhen.
ED73E20F           LD (DATAL),SP    ; SP zum Puffer übertragen.
21B90F             LD HL,LEDL       ; für CONVDI setzen.
11E50F             LD DE,ADDH       ; für CONVDI setzen.
CD7CFA             CALL CONVDI
CD09F9     DLOOP1: CALL DISPL
DD3500             DEC (IX+00)      ; Zeitgeber für Anzeige.
20F8               JR NZ,DLOOP1
DD3502             DEC (IX+02)      ; Zeitgeber für Anzeige.
20F3               JR NZ,DLOOP1
DD3501             DEC (IX+01)      ; Zeitgeber für Bearbeitungs-
20CD               JR NZ,CLOOP1     ; Routine.
FDE1               POP IY           ; CPU-Register umspeichern.
DDE1               POP IX
F1                 POP AF
E1                 POP HL
D1                 POP DE
C1                 POP BC
FB                 EI               ; Interrupt freigeben.
ED4D               RETI             ; von Interrupt zurück.
```

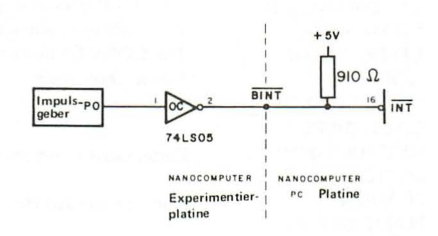


Bild 6-3. Externe Schaltung zur Erzeugung eines aktiven $\overline{\text{INT}}$-Signals.

1. Schritt

Wie nimmt die Z-80-CPU maskierbare Unterbrechungen von externen Geräten an und bestätigt sie? Ebenso wie es einen speziellen CPU-Zyklus zum Lesen und Schreiben von I/Os und Speichern gibt, besteht auch ein spezieller Zyklus für die Anordnung und Bestätigung von Interrupts. Es ist der Z-80-Interrupt-Anforderungs-/Bestätigungszyklus. In Bild 6-4 ist das Zeitdiagramm für diesen Zyklus dargestellt.

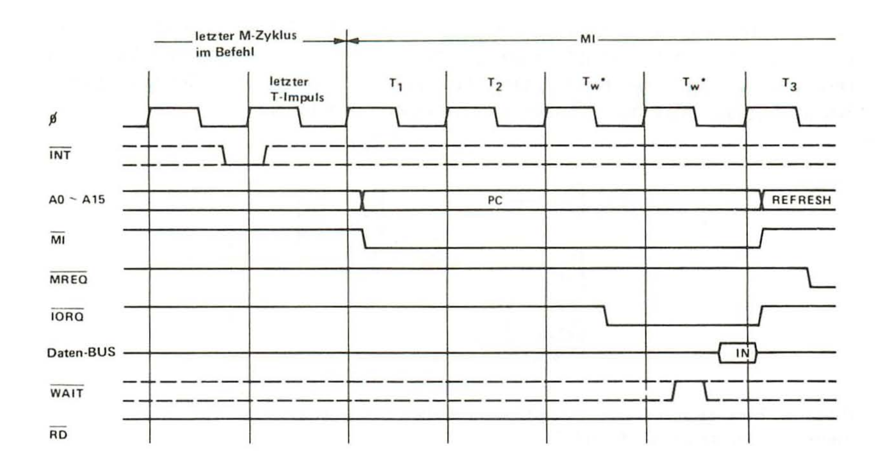


Bild 6-4. Interrupt-Anforderungs-/Bestätigungszyklus der Z-80-CPU.

Die CPU tastet den $\overline{\text{INT}}$-Eingang mit der ansteigenden Flanke des letzten Taktzyklus am Ende aller Befehle ab. Der M-Zyklus im Befehl ist in Bild 6-4 dargestellt. Der logische Zustand am $\overline{\text{INT}}$-Pin spielt nur während der ansteigenden Flanke des letzten T-Impulses vom letzten M-Zyklus des Befehls eine Rolle. Dies ist der einzige Zeitpunkt, wo die CPU diesen Zustand prüft. Wenn das $\overline{\text{INT}}$-Signal aktiv (logisch 0) ist, führt die CPU in der Regel einen Interrupt-Anforderungs-/Bestätigungszyklus aus. Nur in zwei Fällen wird das $\overline{\text{INT}}$-Signal ignoriert:

1. Wenn das $\overline{\text{BUSRQ}}$-Signal aktiv (logisch 0) ist;
2. wenn das Interrupt-Freigabe-Flipflop IFF1 nicht gesetzt ist.

Mit dem Signal $\overline{\text{BUSRQ}}$ kontrollieren die externen Geräte den Adreß-, Daten und Steuer-BUS.

Es sei in diesem und in allen folgenden Versuchen vorausgesetzt, daß $\overline{\text{BUSRQ}}$ nicht aktiv ist. Falls IFF1 zurückgesetzt ist, sind maskierbare Unterbrechungen blockiert.

Bei einem Interrupt-Anforderungs-/Bestätigungszyklus treten einige wichtige Ereignisse ein. In chronologischer Reihenfolge sind dies:

1. Das Signal $\overline{\text{M1}}$ wird aktiv (auf logisch 0 gebracht); dadurch startet ein spezieller M1-Zyklus.
2. Der Inhalt des Programm-Zählregisters gelangt auf den Adreß-BUS.
3. Zwischen T2 und T3 fügen sich automatisch zwei Wartezustände ein.
4. Das Signal $\overline{\text{IORQ}}$wird aktiviert.
5. Nach der Prüfung, ob weitere Wartezustände vorhanden sind, liest die CPU die Daten von dem Daten-BUS ein.

Das Auflegen der PC-Daten auf den Adreß-BUS ist lediglich durch den M1-Zyklus bedingt. Mit T3 beginnt eine Refresh-Operation (auffrischen)! Sind sowohl $\overline{\text{M1}}$ und $\overline{\text{IORQ}}$ aktiv (im Zustand logisch 0), ist das im vorigen Abschnitt definierte $\overline{\text{INTA}}$-Signal aktiviert (logisch 0). Für die Bearbeitungen von Interrupts nach der Methode 1 ist dieses Signal nicht

wichtig. Bei Interrupt-Bearbeitungen nach den Methoden 0 und 2 wird es dazu benutzt, den Geräte-Hinweiscode zum Ablesen durch die CPU auf den Daten-BUS zu legen. Der Grund für die beiden Wartezustände behandelt der Abschnitt Prioritäts-Interrupts in Kapitel 9.

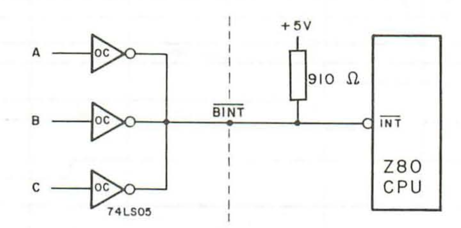


Bild 6-5. Einsatz von Gatter mit offenem Kollektorausgang für den Anschluß mehrerer Geräte am Z-80-$\overline{\text{INT}}$-Pin.

Bauen Sie auf der Experimentierplatine die Schaltung aus Bild 6-5. Zusammen mit dem Impulsgeber P0 kann man über die Inverter die $\overline{\text{BINT}}$-Leitung in den Low-Zustand versetzen und so am Anschluß $\overline{\text{INT}}$ Interrupts erzeugen. Die Nanocomputer®-Platine ist so verdrahtet, daß man an die $\overline{\text{BINT}}$-Leitung Gatter mit offenen Kollektorausgängen anschließen kann. Ohne die Gatter aus Bild 6-5 liegt die $\overline{\text{BINT}}$-Leitung über den 910 Ohm Widerstand an +5 V. Dadurch kann man die in Bild 6-5 gezeigte "Wired-OR"-Schaltung direkt mit der $\overline{\text{BINT}}$-Leitung verbinden.

Alle an den Anschlüssen A, B und C angeschlossenen Geräte können einen Interrupt auslösen, wenn sie den Invertereingang in den Zustand logisch 1 versetzen. Wie bereits erwähnt, müssen die verwendeten Inverter über offene Kollektorausgänge verfügen, um maskierbare Unterbrechungen zu erzeugen. Für den Versuch eignen sich Inverter vom Typ 74LS05. Die Ausgangssignale der angeschlossenen Geräte simuliert der Impulsgeber P0.

Die im Versuch verwendeten drei Software-Routinen sind:

INIT1: Hierbei handelt es sich um eine Initialisierungs-Routine, die folgende Funktion ausübt:

a) Laden des Sprungbefehls an der Speicherstelle 0038H.
b) Setzen des Interrupts auf Methode 1.
c) Einrichten des A'-Registers für einen Abruf an die Subroutine CONVDI.
d) Inhalt von Speicherstelle ADDL in dem dritten und vierten Display (von links nach rechts) anzeigen.

Anmerkung: Bei den Programmen, die mit den Interrupt-Methoden 0 und 1 arbeiten, müssen IM0 und IM1 nicht die erste Anweisung sein. Der Grund dafür ist, daß die GO-Taste die Programmausführung beginnt,indem sie den ersten Programmbefehl nach der Einstufen-Methode ausführt. Da die Einstufen-Version Unterbrechungen nach Methode 2 benutzt, die vom PIO-IC erzeugt werden, bleiben diese zwei Befehle bei der Einstufen-Methode unberücksichtigt. Deshalb erscheinen Sie nicht zuerst in den INIT0- und INIT1-Routinen.

MAIN: Diese Routine führt die Hauptaufgaben durch. Es ist die Routine, die unterbrochen wird. Sie hat folgende Funktionen:

a) Zustandsanzeige des Interrupt-Flipflops IFF1 in der linksbündigen Ziffernstelle der Nanocomputer®-Anzeige.
b) Inhaltsanzeige des Stapelzeiger-Registers (SP) in den rechtsbündigen vier 7-Segment-Anzeigestellen.
c) Verminderung einer Speicherstelle (zwei Verminderungen pro Sekunde), deren Adresse ADDL ist.
d) Inhaltsanzeige der Speicherstelle in der dritten und vierten Anzeigestelle (von links).

SERV1: Dies ist die Programmunterbrechung; sie führt folgende Funktionen aus:
a) Zwischenspeichern der CPU-Register-Inhalte in den Stapelspeicher.
b) Aktualisierung des Daten-Stapelzeigers (Die genaue Beschreibung der Funktion folgt zu einem späteren Zeitpunkt.)
c) Zustandsanzeige des Interrupt-Flipflops IFF1 in der linksbündigen Tastenfeld-Anzeigestelle.
d) Inhaltsanzeige des Stapelzeiger-Registers in den rechtsbündigen vier 7-Segment-Anzeigestellen.
e) Erhöhung des Inhalts der Speicherstelle ADDL um das Zehnfache (ca. 1 Erhöhung pro Sekunde). Dies entspricht genau der halben Geschwindigkeit, mit der die Routine MAIN den Zählwerkinhalt vermindert.
f) Nach jeder Erhöhung Anzeige des Speicherstellen-Inhalts ADDL in der dritten und vierten 7-Segment-Anzeigestelle (von links).
g) Umspeichern der CPU-Register-Inhalte aus dem Stapelspeicher.
h) Steuerungsrückführung zu MAIN.

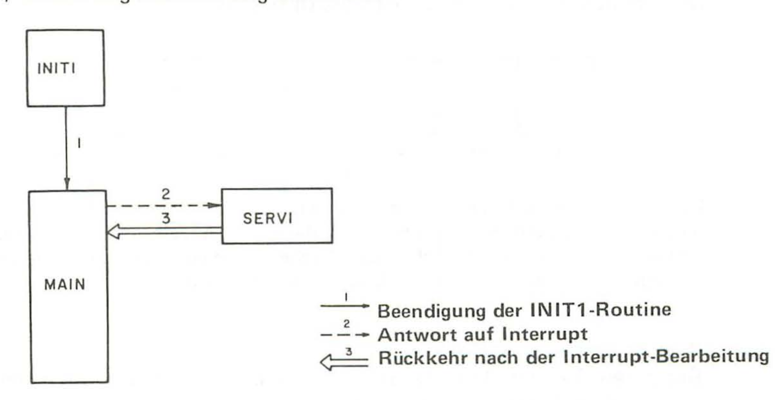


Bild 6-6. Steuerungsablauf zwischen INIT1, MAIN und SERV1.

Der Steuerungsablauf der drei Routinen INIT1, MAIN und SERV1 ist in Bild 6-6 schematisch dargestellt. Der Pfeil 1 symbolisiert die Übertragung der Steuerung zur Routine MAIN nach Beendigung von INIT1. Pfeil 2 zeigt die Übernahme der Steuerung durch SERV1 als Antwort auf einen

Interrupt vom Impulsgeber P0. Pfeil 3 verdeutlicht die Rückkehr der Steuerung von SERV1 nach MAIN nach Beendigung der Programmunterbrechung.

Der Rückkehrbefehl am Ende von SERV1 ist der Befehl RETI! Dieser Befehl steht an Stelle des mormalen RET-Befehls. Er teilt der CPU mit, daß die Rückkehr von einer Programmunterbrechung und nicht von einer CALL-Routine stammt. Die Bedeutung wird am Abschnitt über Prioritäts-Unterbrechungen und Verkettungen (Kapitel 9) erläutert.

Durch das Zusammenwirken von Hard- und Software tritt die nachstehende Ereignisfolge ein, sobald Sie das Programm ab der Speicherstelle INIT1 ausführen:

1. Die Unterbrechungsart wird auf Methode 1 gesetzt, indem der Befehl IM1 abläuft.
2. Das Interrupt-Flipflop IFF1 wird duch Ausführung des Befehls EI freigegeben.
3. Ein externes Gerät aktiviert den $\overline{INT}$-Anschluß.
4. Die CPU empfängt und bestätigt die Unterbrechung durch Aussenden eines $\overline{INTA}$-Signals. Es ist aktiv, wenn $\overline{M1}$ und $\overline{IORQ}$ aktiviert sind. Gleichzeitig wird das Interrupt-Flipflop IFF1 rückgesetzt, um weitere maskierbare Unterbrechungen zu blockieren.
5. Die CPU läuft die Speicherstelle 0038H an, wobei die Steuerung zur Programmunterbrechung übertragen wird.

Beginnen Sie mit der Ausführung von INIT1. Die 7-Segmentanzeigen des Nanocomputers® müssen wie in Bild 6-7 gezeigt aufleuchten. Die Anzeige macht deutlich, daß Interrupt-Flipflop IFF1 auf logisch 1 gesetzt ist, der laufende Vorgang mit ca. zwei Zählungen pro Sekunde gezählt wird und der Inhalt des Stapelzeige-Registers 0F00H ist.

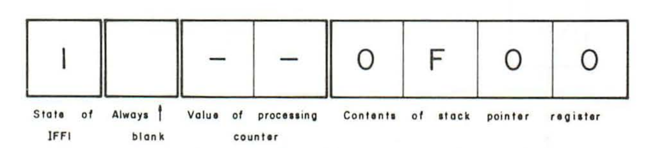


Bild 6-7. Anzeige auf dem 7-Segment-Display. Für die einzelnen Anzeigen gilt (von links): Zustand des Flipflops IFF1; die zweite Anzeige bleibt immer dunkel, Anzeige 3 und 4 geben den Inhalt des Bearbeitungszählers an; auf den letzten vier Anzeigen erscheint der Inhalt des Stapelzeiger-Registers.

3. Schitt

Betätigen Sie den Impulsgeber P0. Dadurch geht der $\overline{INT}$-Eingang kurzzeitig auf logisch 0. Was beobachten Sie?

a) Das Interrupt-Flipflop IFF1 wird zurückgesetzt; auf der ersten Anzeige erscheint eine 0. Durch den Empfang und die Bestätigung sind weitere Interrupts blockiert.

b) Das Zählregister vermindert nicht mehr, sondern erhöht seinen Inhalt. Die Zählgeschwindigkeit in der Interrupt-Routine ist gegenüber dem normalen Betrieb um etwa die Hälfte langsamer. Für diese Änderungen ist der Programmblock SERV1 verantwortlich, der durch den Interrupt-Impuls die Steuerung übernommen hat.

c) Das Stapelzeiger-Register kann zwei verschiedene Inhalte anzeigen:
Fall 1: 0EF2 (falls die MAIN-Routine beim Beginn der Unterbrechung keine der beiden Subroutinen CONVDI oder DISPL ausführte) oder
Fall 2: 0EEE (falls die MAIN-Routine beim Beginn der Unterbrechung eine der beiden Subroutinen CONVDI oder DISPL ausführte).

d) Nach 10 Sekunden ist der Interrupt-Vorgang beendet. Die Anzeige kehrt zu ihrem ursprünglichen Verhalten zurück. Die Wahrscheinlichkeit, daß zu Beginn der Unterbrechung entweder CONVDI oder DISPL läuft, ist relativ hoch. Nachstehend eine Analyse, warum die Zeitfolge der Unterbrechung das Stapelzeiger-Register beeinflußt:

Analyse zu Fall 1: Die Routine SERV1 schiebt sechs 2-Byte-Worte (6 Registerpaare bzw. 16-Bit-Register) auf den Stapelspeicher. Die CPU schiebt beim Wiederanlauf zur Speicherstelle 0038 zwei weitere Bytes von der Rücksprungadresse hinzu. So sind insgesamt sieben 2-Bytepaare oder 14 Bytes während der Unterbrechung in den Stapelspeicher gelangt. Der hexadezimale Wert der Ziffer 14 ist 000E. Die hexadezimale Addition von 0EF2 und 000E ergibt 0F00.

Analyse zu Fall 2: CONVDI und DISPL schieben genau ein Registerpaar in den Stapelspeicher. Ein weiteres 2-Byte-Wort kommt von der Rücksprungadresse durch den Subroutinen-Aufruf CONVDI oder DISPL hinzu. Das ergibt zusammen mit den sieben 2-Byte-Worten von der SERV1-Ausführung 18 Bytes (hex 12), die zum Stapelspeicher hinzugefügt werden. Die Assition von 0EEE und 0012 ergibt 0F00!

4. Schritt

Erzeugen Sie mit dem Impulsgeber P0 ein Interrupt-Signal. Während das Flipflop IFF1 zurückgesetzt ist, geben Sie ein zweites Interrupt-Signal zur CPU. Das zweite Signal an $\overline{INT}$ darf die Anzeige nicht verändern, weil IFF1 zurückgesetzt ist. Dadurch sind maskierbare Interrupts blockiert.
Der nächste Schritt zerstört das gespeicherte Programm. Es ist daher ratsam, diesen Schritt nur zu lesen und nicht auszuführen.

5. Schritt

An der Speicherstelle DS1 + 6 in der SERV1-Routine befindet sich ein NOP-Befehl (hex. 00). Ändern Sie diesen NOP-Befehl in den Interrupt-Freigabebefehl EI (hex. FB). Führen Sie das Programm aus, beginnend bei INIT1. Aktivieren Sie das $\overline{INT}$-Signal durch Schalten des Impulsgebers P0. Was beobachten Sie?

Die Anzeige erlischt. Setzen Sie den Nanocomputer® zurück und prüfen Sie den Speicher ab Speicherstelle DS1 + 6. Der ursprüngliche Programmcode ist zerstört. Was hat sich ereignet? Für das weitere Verständnis ist eine Tatsache sehr wichtig: *Das $\overline{INT}$-Signal ist pegelempfindlich.* Die Z-80-CPU tastet den Zustand den $\overline{INT}$-Pin mit der ansteigenden Flanke des letzten Taktzyklus am Ende aller Befehle ab. Ist IFF1 freigegeben (also gesetzt) und $\overline{BUSRQ}$ nicht aktiv, beginnt die Programmunterbrechung; in diesem Fall SERV1. Als Folge werden mehrere Bytes auf den Stapelspeicher geschoben. Die Änderung von NOP in EI ist sehr kritisch. Der

Befehl EI löst eine Programmunterbrechung aus, in der maskierbare Interrupts früh freigegeben werden. Gibt der Impulsgeber P0 einen Interruptimpuls ab, ist das $\overline{INT}$-Signal für die Zeit von mehreren 100 Befehlen logisch 0. Demzufolge bearbeitet die Z-80-CPU mehrere hundert Interrupt-Anforderungen, bevor der Impulsgeber in seine Ruhestellung zurückkehrt. (Addiert man die Befehlsausführungszeiten, wird der EI-Befehl nur 50 Mikrosekunden nach dem Abtasten des $\overline{INT}$-Pins durch die CPU ausgeführt). Bei jeder Interrupt-Bearbeitung dehnt sich der Stapelspeicher aus, bis er schließlich das laufende Programm überschreibt.
Bei der ursprünglichen Version der SERV1-Routine ist das Problem der Pegelempfindlichkeit nicht relevant. Maskierbare Unterbrechungen sind dabei bis zur Beendigung der Interrupt-Bearbeitung blockiert. Das ist nicht immer eine wünschenswerte Lösung. Es ist daher ratsam, wenn der Impulsgeber die $\overline{INT}$-Leitung nicht direkt aktiviert, sondern die flankengetriggerte Takteingabe eines Flipflops steuert. Dadurch erscheint das Interrupt-Signal von einem externen Gerät (z.B. Impulsgeber) als ein *flankengetriggertes Signal*. Bild 6-8 zeigt eine Schaltung für ein flankengetriggertes Interrupt-Signal. Das Flipflop 74LS74 kopiert den logischen Zustand am Eingang D zum Ausgang Q, wenn folgende Bedingungen erfüllt sind: CD- und SD-Eingang müssen logisch 1 sein und am Takteingang ein positiver Impuls anstehen. Am $\overline{Q}$-Ausgang des Flipflops ist der Logikpegel von Eingang D invertiert. Die Schaltung aus Bild 6-8 aktiviert also mit einem positiven Taktimpuls den $\overline{INT}$-Eingang der CPU. Hat die CPU die Unterbrechung bestätigt, geht das $\overline{INTA}$-Signal auf logisch 0. Auch der Ausgang des AND-Gatters 74LS08 geht auf logisch 0, wodurch der Flipflop-Ausgang $\overline{Q}$ logisch 0 wird. Der Inverter OC invertiert das Signal, so daß der $\overline{INT}$-Eingang wieder inaktiv ist. Sebst wenn jetzt noch weitere Taktimpulse zum Flipflop gelangen, ändert sich dessen Zustand nicht, solange der $\overline{SD}$-Eingang low bleibt.
Es gibt sicherlich noch viele andere Lösungen, den $\overline{INT}$-Eingang nur kurzzeitig im Zustand low zu halten. Die Schaltung in Bild 6-8 arbeitet als bistabiler Multivibrator. Die bistabile Version erfüllt nicht nur die gewünschte Funktion, sondern macht ebenfalls die Arbeitsweise der internen Flipflops bei z.B. den PIO- und CTC-ICs deutlich. Bei nichtmaskierbaren Interrupt-Anforderungen wird der monostabile Multivibrator verwendet; die Funktion ist ähnlich wie bei einem Impulsgeber.

6. Schritt
Verdrahten Sie die in Bild 6-8 dargestellte Schaltung. Falls Sie Schritt 5 praktisch ausgeführt haben, müssen Sie das Programm überprüfen und evt. neu eingeben.
Ändern Sie den NOP-Befehl an der Speicherstelle DS1 + 6 auf EI (hex FB). Starten Sie das Programm bei INIT1. Betätigen Sie den Impulsgeber P0, damit der Takteingang des Flipflops von logisch 0 nach logisch 1 wechselt. Belassen Sie den Schalter in dieser Stellung. Was beobachten Sie?
Der Nanocomputer® bearbeitet nur eine Interrupt-Anforderung.

7. Schritt
Ändern Sie den Befehl bei DS1 + 6 von EI wieder auf NOP. Führen Sie das Programm ab INIT1 erneut aus. Betätigen Sie den Impulsgeber. Während

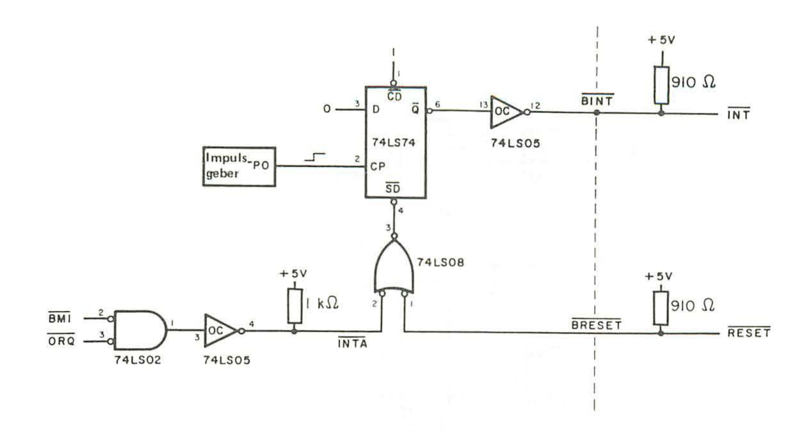


Bild 6-8. Schaltung für Interrupt-Vorbereitungs-Flipflop.

die Programmunterbrechung ausgeführt wird, Impulsgeber nochmals schalten. Was beobachten Sie?
Die Bearbeitung beider Unterbrechungen folgt ohne sichtbare Rückkehr der Steuerung zur MAIN-Routine. Das Flipflop (Bild 6-8) speichert die zweite Unterbrechung, bevor der EI-Befehl am Ende der SERV1-Routine die CPU veranlaßt, sie anzunehmen und zu bestätigen. Das zweite $\overline{\text{INTA}}$-Signal setzt das Flipflop selbstverständlich zurück. Die Speichereigenschaft verleiht dem Flipflop seinen Namen.

8. Schritt

Impulsgeber betätigen. Während die Programmunterbrechung ausgeführt wird, Impulsgeber noch einigemale schalten. Was beobachten Sie?
Es werden lediglich nur zwei Unterbrechungen nacheinander bearbeitet, weil ein Flipflop nur eine bevorstehende Unterbrechung speichern kann.

LASSEN SIE DIE SCHALTUNG AUS BILD 6-8 BESTEHEN, SIE WIRD FÜR DEN NÄCHSTEN VERSUCH BENÖTIGT!

VERSUCH NR. 2

Der Versuch macht Sie mit der Z-80-Interrupt-Methode 0 bekannt. Diese Methode der Unterbrechungsbearbeitung ist identisch mit der des Mikroprozessors 8080A.

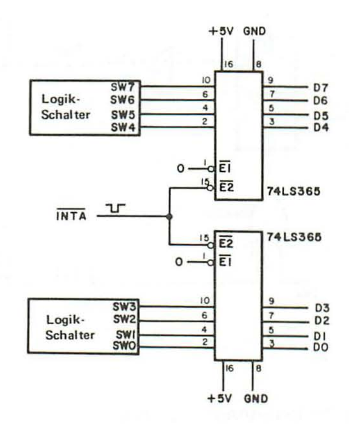


Bild 6-9.

Zusätzlich zur Schaltung aus Bild 6-8 benötigen Sie für diesen Versuch noch die Schaltung aus Bild 6-9.

Programme INITO

Neben den zwei Programmen MAIN und SERV1 aus Versuch Nr. 1 wird noch das folgende Programm ausgeführt:

Objekt-Code		Quell-Code	Bemerkung
		NAME INITO	
3EC3	INITO:	LD A,0C3H	; erstes Byte ist ein Sprung.
323800		LD(0038H),A	; in Speicherstelle RST laden
FD216E02		LD IY,SERV1	; Adresse der Bearbeitungs-Routine Nr. 1.
FD223900		LD (0039H),IY	
ED46		IMO	; Interrupt-Modus 0.
08		EX AF,AF'	: Leerstellen in der Anzeige festsetzen.
3E40		LD A,40H	; für CONVDI setzen.
08		EX AF,AF'	
C3C302		JP MAIN	; zurück zur MAIN-Routine.

1. Schritt

Der Interrupt-Anforderungs/Bestätigungszyklus für die beiden Unterbrechungsmethoden (Z-80-Interrupt-Modus 0 und 1) ist genau gleich. Der wichtigste Unterschied zwischen den beiden besteht darin, was die CPU vom Interrupt-Gerät erwartet. Bei der Unterbrechungsbearbeitung nach Methode 1 genügt ein low-aktives Anforderungssignal vom Interrupt-Gerät. Bei der Unterbrechungsbearbeitung nach der Methode 0 erwartet die CPU

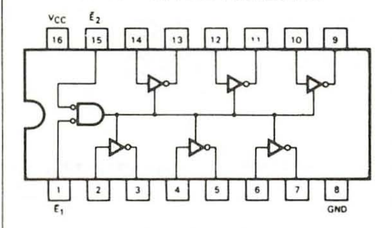
Tabelle 6-3. Eigenschaften des ICs 74LS365

T54LS365/T74LS365
HEX 3-STATE BUFFER WITH
COMMON 2-INPUT NOR ENABLE

TRUTH TABLE

INPUTS			OUTPUT
$\bar{E}_1$	$\bar{E}_2$	D	
L	L	L	L
L	L	H	H
H	X	X	(Z)
X	H	X	(Z)

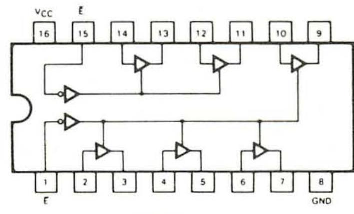
T54LS366/T74LS366
HEX 3-STATE INVERTER BUFFER
WITH COMMON 2-INPUT NOR ENABLE



TRUTH TABLE

INPUTS			OUTPUT
$\bar{E}_1$	$\bar{E}_2$	D	
L	L	L	H
L	L	H	L
H	X	X	(Z)
X	H	X	(Z)

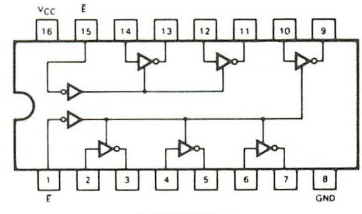
T54LS367/T74LS367
HEX 3-STATE BUFFER
SEPARATE 2-BIT AND 4-BIT SECTIONS



TRUTH TABLE

INPUTS		OUTPUT
$\bar{E}$	D	
L	L	L
L	H	H
H	X	(Z)

T54LS368/T74LS368
HEX 3-STATE INVERTER BUFFER
SEPARATE 2-BIT AND 4-BIT SECTIONS



TRUTH TABLE

INPUTS		OUTPUT
$\bar{E}$	D	
L	L	H
L	H	L
H	X	(Z)

DESCRIPTION — The LS365/366/367/368 are high speed hex buffers with 3-state outputs. They are organized as single 6-bit or 2-bit/4-bit, with inverting or non-inverting data (D) paths. The outputs are designed to drive 15 TTL Unit Loads or 60 Low Power Schottky loads when the Enable ($\bar{E}$) is LOW.

When the Output Enable Input ($\bar{E}$) is HIGH, the outputs are forced to a high impedance "off" state. If the outputs of the 3-state devices are tied together, all but one device must be in the high impedance state to avoid high currents that would exceed the maximum ratings. Designers should ensure that Output Enable signals to 3-state devices whose outputs are tied together are designed so there is no overlap.

Tabelle 6-3. Fortsetzung

GUARANTEED OPERATING RANGES

PART NUMBERS		SUPPLY VOLTAGE MIN	SUPPLY VOLTAGE TYP	SUPPLY VOLTAGE MAX	TEMPERATURE
T54LS365X T54LS367X	T54LS366X T54LS368X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS365X T74LS367X	T74LS366X T74LS368X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type, D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.4	3.4			I_{OH} = −1.0 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.4	3.1			I_{OH} = −2.6 mA	
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 12 mA	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74		0.35	0.5	V	I_{OL} = 24 mA	
I_{OZH}	Output Off Current HIGH				20	μA	V_{CC} = MAX, V_{OUT} = 2.4 V, $V_{\bar{E}}$ = 2.0 V	
I_{OZL}	Output Off Current LOW				−20	μA	V_{CC} = MAX, V_{OUT} = 0.4 V, $V_{\bar{E}}$ = 2.0 V	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Output Short Circuit Current (Note 3)		−30		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V	
I_{CC}	Power Supply Current	LS365/367		13.5	24	mA	V_{CC} = MAX, V_{IN} = 0 V, $V_{\bar{E}}$ = 4.5 V	
		LS366/368		11.8	21			

NOTES
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

AC CHARACTERISTICS: T_A = 25°C, V_{CC} = 5.0 V (See Page 4-41 for Waveforms)

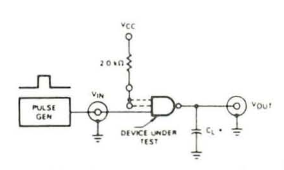
SYMBOL	PARAMETER	LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS	
t_{PLH} t_{PHL}	Propagation Delay, Data to Output (LS365 • LS367)			10 16	ns	Fig. 2	C_L = 45 pF
t_{PLH} t_{PHL}	Propagation Delay, Data to Output (LS366 • LS368)			10 16	ns	Fig. 1	C_L = 45 pF
t_{PZH}	Output Enable Time to HIGH Level			16	ns	Figs. 4, 5	C_L = 45 pF
t_{PZL}	Output Enable Time to LOW Level			30	ns	Figs. 3, 5	R_L = 667 Ω
t_{PLZ}	Output Disable Time from LOW Level			15	ns	Figs. 3, 5	C_L = 5.0 pF
t_{PHZ}	Output Disable Time from HIGH Level			23	ns	Figs. 4, 5	R_L = 667 Ω

Tabelle 6-3. Fortsetzung

AC TEST CIRCUITS AND WAVEFORMS

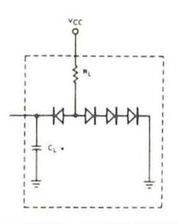
The following test circuits and conditions represent SGS-ATES's typical AC test procedures. The output loading for standard Low Power Schottky devices is a 15 pF capacitor. Experimental evidence shows that test results using the additional diode-resistor load are within 0.2 ns of the capacitor only load. The capacitor only load also has the advantage of repeatable, easily correlated test results. The input pulse rise and fall times are specified at 6 ns to closely approximate the Low Power Schottky output transitions through the active threshold region. The specified propagation delay limits can be guaranteed with a 15 ns input rise time on all parameters except those requiring narrow pulse widths. Any frequency measurement over 15 MHz or pulse width less than 30 ns must be performed with a 6 ns input rise time.

Test Circuit for Standard Output Devices

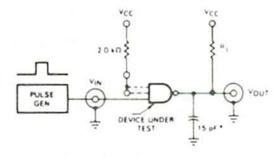


*Includes all probe and jig capacitance

Optional Load (Guaranteed–Not Tested)



Test Circuit for Open Collector Output Devices



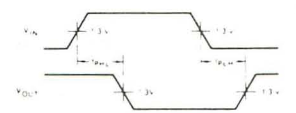
*Includes all probe and jig capacitance

Pulse Generator Settings
(unless otherwise specified)

Frequency = 1 mHz
Duty Cycle = 50%
t_{TLH} (t_r) = 6 ns
t_{THL} (t_f) = 6 ns
Amplitude = 0 to 3 V



Waveform for Inverting Outputs



Waveform for Non-inverting Outputs

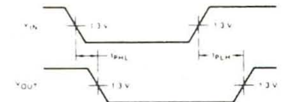
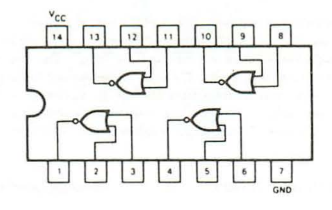


Tabelle 6-4. Eigenschaften des ICs 74LS02

QUAD 2-INPUT NOR GATE



GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS02X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS02X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.4		V	V_{CC} = MIN, I_{OH} = −400 µA, V_{IN} = V_{IL}
		74	2.7	3.4			
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 10 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current (Note 3)		−15		−100	mA	V_{CC} = MAX, V_{OUT} = 0 V
I_{CCH}	Supply Current HIGH			1.6	3.2	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			2.4	5.4	mA	V_{CC} = MAX, Inputs Open

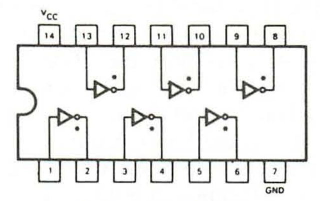
AC CHARACTERISTICS: T_A = 25°C (See Page 4-50 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output	3.0	5.0	10	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output	3.0	5.0	10	ns	C_L = 15 pF

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.
3. Not more than one output should be shorted at a time.

Tabelle 6-5. Eigenschaften des ICs 74LS05

HEX INVERTER



*OPEN COLLECTOR OUTPUTS

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE			TEMPERATURE
	MIN	TYP	MAX	
T54LS05X	4.5 V	5.0 V	5.5 V	−55°C to 125°C
T74LS05X	4.75 V	5.0 V	5.25 V	0°C to 75°C

X = package type; D for Ceramic Dip, B for Plastic Dip. See Packaging Information Section for packages available on this product.

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS			UNITS	TEST CONDITIONS (Note 1)
			MIN	TYP	MAX		
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage
		74			0.8		
V_{CD}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
I_{OH}	Output HIGH Current				100	µA	V_{CC} = MIN, V_{OH} = 5.5 V, V_{IN} = V_{IL}
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	V_{CC} = MIN, I_{OL} = 4.0 mA, V_{IN} = 2.0 V
		74		0.35	0.5	V	V_{CC} = MIN, I_{OL} = 8.0 mA, V_{IN} = 2.0 V
I_{IH}	Input HIGH Current			1.0	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 5.5 V
I_{IL}	Input LOW Current				−0.36	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{CCH}	Supply Current HIGH			1.2	2.4	mA	V_{CC} = MAX, V_{IN} = 0 V
I_{CCL}	Supply Current LOW			3.6	6.6	mA	V_{CC} = MAX, Inputs Open

AC CHARACTERISTICS: T_A = 25°C (See Page 4-50 for Waveforms)

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH}	Turn Off Delay, Input to Output		14	22	ns	V_{CC} = 5.0 V
t_{PHL}	Turn On Delay, Input to Output		10	18	ns	C_L = 15 pF, R_L = 2.0 kΩ

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, T_A = 25°C.

vom externen Gerät außer dem Anforderungssignal noch ein 8-Bit-Befehls-Byte. Dieses Befehls-Byte kann ein Ein- oder Mehrfach-Byte-Befehl sein. Die von Ihnen für diesen Versuch benutzte Schaltung ist nur für die Handhabung eines Ein-Byte-Befehls konzipiert. Die CPU bedient einen Interrupt nach Methode 0, indem sie den vom externen Gerät erteilten Befehl liest, dekodiert und ausführt. In der Praxis sind die allgemein ausgeführten Befehle Ein-Byte-RST-Befehle, da sie höchste Flexibilität gewährleisten und als Subroutine-Abrufe den Inhalt des PC-Registers zwischenspeichern. Wenn das Interrupt-Gerät das Byte FF sendet – das ist der Operations-Code für den Befehl RST 38H – ist das Ergebnis mit dem der Unterbrechung nach Methode 1 identisch.
Bild 6-9 zeigt die Schaltung, die Sie in Verbindung mit dem Impulsgeber P0 und der Schaltung aus Bild 6-8 für die Versuche mit der Unterbrechungsbearbeitung nach Methode 0 benötigen. Der Impulsgeber sendet das Interrupt-Anforderungssignal zur CPU, während die Schaltung aus Bild 6-9 einen Ein-Byte-Befehl im richtigen Augenblick beim Unterbrechungs-Anforderungs/Bestätigungszyklus auf den Daten-BUS legt. Um die logischen Schaltdaten für den Daten-BUS freizugeben, wird das Signal $\overline{\text{INTA}}$ benutzt. Wenn das $\overline{\text{INTA}}$-Signal nicht aktiv ist, nehmen die Puffer einen hochohmigen Zustand ein.
Bauen Sie die in Bild 6-9 gezeigte Schaltung und verbinden sie mit der Schaltung aus Bild 6-8. Achten Sie darauf, daß der Impulsgeber P0 mit dem Flipflop 74LS74 verbunden ist. Verdrahten Sie außerdem die LED-Anzeige aus Bild 6-10. Diese Schaltung ist ein LED-Monitor für die $\overline{\text{HALT}}$-Leitung. Sie ist für Unterbrechungen nach Methode 0 nicht erforderlich, wird aber später noch benutzt.

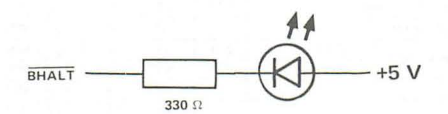


Bild 6-10. Schaltung für zusätzlichen LED-Monitor zur Kontrolle von $\overline{\text{BHALT}}$-gepufferten CPU-$\overline{\text{HALT}}$-Signalausgaben.

2. Schritt

Die in diesem Versuch benutzte Software besteht aus drei Routinen: INIT0, MAIN und SERV1. Die Funktionen von MAIN und SERV1 sind bereits im ersten Versuch dieses Kapitels beschrieben. Die Routine INIT0 ist identisch mit INIT1 aus Versuch Nr. 1, nur setzt sie die Interrupt-Bearbeitung nach Methode 0 anstatt nach Methode 1.
Der Steuerungsablauf zwischen den Routinen INIT0, MAIN und SERV1 geht aus Bild 6-11 hervor.
Alle Logik-Schalter in ON-Stellung bringen, damit sie hex FF lesen. Das Befehls-Byte FF ist der Operations-Code für den Befehl RST 38H, durch den die CPU einen Subroutinesprung zur Speicherstelle 0038 ausführt. Führen Sie das Programm ab Speicherstelle INIT0 aus. Das Display zeigt die gleichen Daten wie im ersten Versuch (Bild 6-7).

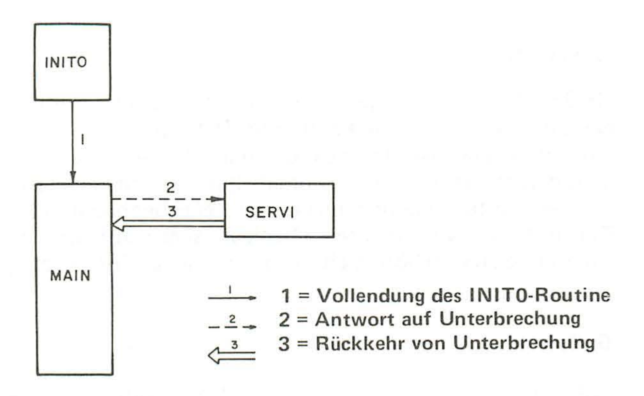


Bild 6-11. Steuerungsablauf zwischen den Routinen INITO, MAIN und SERV1.

3. Schritt

Erzeugen Sie durch Schalten des Impulsgebers P0 eine Unterbrechung. Das Ergebnis ist gleich wie beim Interrupt nach Methode 1 im ersten Versuch. Stoppen Sie den Programmablauf und nehmen folgende Änderung in der Routine MAIN vor.

Speicherstelle	*Inhalt*	*umändern in*
DISAB	00	F3 (Unterbrechungen blockieren)

Diese Änderung blockiert das Interrupt-Flipflop IFF1, bis der EI-Befehl an Speicherstelle MAIN erneut ausgeführt ist. Dadurch wird ein Interrupt nicht angenommen, wenn die Subroutine CONVDI oder DISPLY abläuft. Die Z-80-Unterbrechung nach Methode 0 ermöglicht es, einzelne Befehle (z.B. Stapelzeiger (Stack-Pointer) erhöhen) auf den Daten-BUS zu legen und die Befehle auszuführen. Daher darf bei der Ausführung einer Subroutine die CPU vom Programm MAIN keine Interrupts annehmen.

Anmerkung: Arbeiten Programme mit der Interrupt-Methode 0 oder 1, muß der Befehl IM0 oder IM1 nicht die erste Anweixung sein. In beiden Fällen beginnt das Programm nach dem Drücken der GO-Taste nach der Einstufen-Methode. Dabei werden vom PIO-IC Interrupts nach Methode 2 erzeugt.

4.Schritt

Stellung der Logik-Schalter von FF in 76 ändern. 76 ist der Operations-Code für den HALT-Befehl. Erzeugen Sie eine Unterbrechung. Was beobachten Sie?

Die Anzeige auf den 8 Displays erlischt und die Anzeige-LED für das $\overline{\text{HALT}}$-Signal leuchtet auf. Der Z-80 wechselt bei einem HALT-Befehl in den HALT-Zustand über. Dadurch führt die CPU zwangsläufig soviele NOP-Befehle aus, bis ein aktiviertes RESET-Signal gefunden ist.

5. Schritt

RESET-Taste betätigen, um die Steuerung zum Betriebssystem des Nanocomputers® zurückzuführen. Mit den Logik-Schaltern den Befehl 33 einstellen. Das ist der hexadezimale Code für den Befehl INC SP. Programmausführung noch einmal bei der Speicherstelle INITO starten. Mehrere Unterbrechungen erzeugen. Was beobachten Sie?
Der mit den vier rechten Displays angezeigte Inhalt des Stack-Pointers (Stapelregister) erhöht sich mit jedem neuen Interruptsignal einmal.

6. Schritt

Mit den Logik-Schaltern den Operations-Code 3B einstellen; das entspricht dem Befehl DEC SP. Was ändert sich jetzt mit jedem Interrupt-Signal?
Mit jedem Interrupt-Signal wird der Stack-Pointerinhalt einmal vermindert. Bei diesem und dem 4. Schritt scheint der Zähler ohne Unterbrechung weiterzuzählen. Das Interrupt-Flipflop IFF1 wechselt nur kurz von 1 nach logisch 0, sobald die CPU die Unterbrechung erkennt und bestätigt. Während der Befehl EI auf der Speicherstelle MAIN ausgeführt wird, wechselt das Flipflop wieder nach logisch 1. Bei der Befehlsänderung von 33 nach 3B braucht man nur einen Logik-Schalter umzustellen, nämlich SW3. Es ist somit relativ einfach, durch Ändern von SW3 und Erzeugung von Interrupt-Signalen den Inhalt des Stack-Pointers positiv oder negativ zu beeinflussen.

7. Schritt

Die Logik-Schalter auf C5 einstellen; das ist der Operations-Code für PUSH BC. Welche Änderung zeigt das Display bei einem Interrupt-Signal?
Der Inhalt des Stack-Pointers wird um zwei vermindert. Das ist eine logische Folge des Befehls PUSH BC, der in den Stack-Pointer zwei Bytes ablegt.

8. Schritt

Mit den Logik-Schaltern den Befehl NOP (00H) einstellen. In diesem Fall beeinflussen die Interrupt-Signale nur die IFF1-Anzeige. Sie wechselt kurz von logisch 1 nach logisch 0.
Wählen Sie andere Ein-Byte-Befehle aus und geben Sie diese mit Hilfe der Logik-Schalter auf den Daten-BUS. Erzeugen Sie bei jedem neuen Befehl Interrupts, welche die CPU wie bisher nach der Methode 0 verarbeitet. Bestimmen Sie vor jedem praktischen Test das Ergebnis und vergleichen es mit der veränderten Anzeige des Nanocomputers®. Begründen Sie das Ergebnis.

Achtung: Die Schaltungen aus diesem Versuch werden auch in Versuch Nr. 3 benötigt.

VERSUCH NR. 3

Der Versuch macht Sie mit der Interrupt-Methode 2 bekannt. Für diesen Versuch sind die Schaltungen aus den Bildern 6-8, 6-9 und 6-10 erforderlich.

Programme INIT2, SERV2 und SERV3

Dieser Versuch benötigt zu den Routinen MAIN und SERV1 noch die folgenden Programme:

Objekt-Code		Quell-Code	Bemerkung
		NAME INIT2	
ED5E	INIT2:	IM2	; Interrupt-Methode 2.
21000F		LD HL,TABLE	; Adresse der Vektor-Tabelle.
7C		LD A,H	; High-Byte der Adresse.
ED47		LD I,A	; Interrupt-Register setzen.
ED216E02		LD IY,SERV1	; Erste Bearbeitungsroutine.
FD22000F		LD (TABLE),IY	; Vektor-Tabelle einsetzen.
FD21F502		LD IY,SERV2	; Zweite Bearbeitungsroutine.
FD22020F		LD (TABLE+2),IY	; Vektor-Tabelle einsetzen.
FD216B03		LD IY,SERV3	; Dritte Bearbeitungsroutine.
FD22040F		LD (TABLE+4),IY	; Vektor-Tabelle einsetzen.
08		EX AF,AF'	, CONVDI setzen.
3E40		LD A,40H	
08		EX AF,AF'	
C3C302		JP MAIN	; Zur MAIN-Routine springen.

Objekt-Code		Quell-Code	Bemerkung
		NAME SERV2	
76	SERV2:	HALT	; Mikrocomputer stoppen

Objekt-Code		Quell-Code	Bemerkung
		NAME SERV3	
C5	SERV3:	PUSH BC	; CPU-Registerinhalt zwischen-
D5		PUSH DE	; speichern.
E5		PUSH HL	
F5		PUSH AF	
DDE5		PUSH IX	
FDE5		PUSH IY	
DD23	DS3:	INC IX	; Daten-Stapelspeicher
DD23		INC IX	; aktualisieren (Stack-Pointer).
DD23		INC IX	
00		NOP	; keine Operation.
DD3600FF		LD(IX+00H),0FFH	; DLOOP3-Zeit setzen.
DD36010A		LD(IX+01H),00AH	; CLOOP3-Zeit setzen.
DD360202	CLOOP3:	LD(IX+02H),02H	; DLOOP3-Zeit setzen.
21E50F		LD HL,ADDH	; Hinweis auf Anzeige-Puffer.
ED57		LD A,I	; IFF2-Wert suchen
EA9203		JP PE,HIGH3	
3600	LOW3:	LD (HL),00H	; Wert = 0
1802		JR NEXT3	
3610	HIGH3:	LD (HL),10H	; Wert = 1
2B	NEXT3:	DEC HL	; Puffer-Hinweis verschieben
34		INC (HL)	; ADDL erhöhen.
34		INC (HL)	; ADDL erhöhen.
ED73E20F		LD (DATAL),SP	; SP-Inhalt zum Puffer über-
21B90F		LD HL,LEDL	; tragen.
11E50F		LD DE,ADDH	; CONVDI setzen.
CD7CFA		CALL CONVDI	; CONVDI setzen.
CD09F9	DLOOP3:	CALL DISPL	
DD3500		DEC (IX+00)	; Zeitgeber für Anzeige.
20F8		JR NZ,DLOOP3	
DD3502		DEC (IX+02)	; Zeitgeber für Anzeige.
20F3		JR NZ,DLOOP3	
DD3501		DEC (IX+01)	; Zeitgeber für Service-Routine.
20CC		JR NZ,CLOOP3	
FDE1		POP IY	; Inhalt der CPU-Register
DDE1		POP IX	; umspeichern.
F1		POP AF	
E1		POP HL	
D1		POP DE	
C1		POP BC	
FB		EI	; Interrupts freigeben.
ED4D		RETI	; Vom Interrupt zurückkehren.

1. Schritt
Bauen Sie die Schaltungen für diesen Versuch – sofern noch nicht geschehen – auf. Der Modus 2 ist die wirkungsvollste Interrupt-Methode. Während die zur Durchführung erforderliche Hardware dieselbe ist wie bei

der Interrupt-Bearbeitungsmethode 0, ist die Software völlig anders. Die Hardware ist deshalb ähnlich, weil die CPU bei der Methode 2 einen 8-Bit-Vektor oder Erkennungscode vom externen Gerät erwartet. Dieser 8-Bit-Code wird aber nicht wie bei Methode 0 als Befehls-Operations-Code übersetzt, sondern vielmehr als das niedrigstwertige Byte einer 16-Bit-Speicheradresse.

Die Bildung der Adresse erfolgt intern durch die Verkettung des Inhalts vom I-Register (für Unterbrechung) mit dem Byte des Interrupt-Gerätes. Bild 6-12 zeigt, wie das I-Register und das Vektor- oder Erkennungs-Byte des externen Geräts vom Daten-BUS sich zu einer 16-Bit-Adresse zusammenschließen, die man *Vektor-Tabellenadresse* nennt. Für alle externen Z-80-ICs, besonders das CTC- und PIO-IC, muß das niedrigstwertige Bit

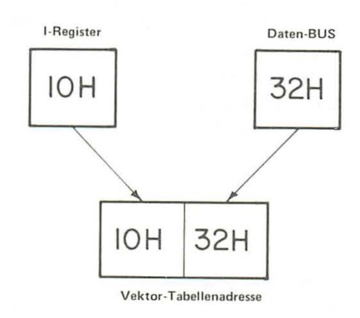


Bild 6-12. Bildung einer Vektor-Tabellenadresse.

dieser Adresse im Zustand logisch 0 sein. Es ist daher die Regel, alle Interrupt-Adressen der Methode 2 auf gerade Speicherstellen auszurichten. Hardware und Software müssen also aufgrund ihrer Konzeption von den externen Geräten *gerade* Low-Bytes erwarten.

Die durch das I-Register und das Erkennungs-Byte gebildete 16-Bit-Adresse wird von der CPU (bei Methode 2) als Hinweisadresse für die Programmunterbrechung übersetzt. Wenn z.B. der Inhalt des I-Registers 10 ist und das externe Gerät das Erkennungs-Byte 32 liefert, ist die Vektor-Tabellenadresse 1032. Die so definierte und die nachfolgende Speicheradresse enthalten das nieder- und höherwertige Byte der Programm-Unterbrechungsadresse. Haben diese Speicherstellen folgenden Inhalt

Speicherstelle	Inhalt
1032	70
1033	02

ist die erste Befehlsadresse für die Programmunterbrechung 0270. In diesem Falle führt die CPU einen Subroutine-Abruf zur Speicherstelle 0270 durch und speichert den Inhalt des PC-Registers auf dem Stapelspeicher (Stack-Pointer) ab. Ein RETI-Befehl gibt die Steurung zum

unterbrochenen Programm zurück.
Bei vielen Interrupt-Bearbeitungen nach Methode 2 wird das I-Register auf einen Wert (z.B. XY) eingestellt und nie verändert. Dieser Wert dient als eine Hinweisadresse auf einen zusammenhängenden Speicherblock von 256 Bytes, nämlich die Speicherstellen XY00 bis XYFF. Diesen Speicherblock nennt man *Interrupt-Vektortabelle,* weil sein Inhalt auf die Herkunft von Programmunterbrechungen hinweist. Da jedes geradzahlig ausgerichtete Paar (XY00-XY01, XY02-XY03 ... XYFE-XYFF) potentiell die Adresse für eine Programmunterbrechung ist, gibt es für einen einzigen I-Registerwert 128 mögliche Interrupts nach Methode 2, welche der Z-80 anerkennt. Falls das nicht genügt, kann man den I-Register-Inhalt auch verändern! Für die meisten Anwendungen sind dies jedoch genügend Interrupt-Möglichkeiten. Im Hinblick auf den Interrupt-Aufbau kann der Z-80 mit anderen Mikroprozessoren durchaus konkurriern. Der Preis für diese Flexibilität und Leistung beträgt 256 Bytes im adressierbaren Speicher. Da der Interrupt-Anforderungs/Bestätigungszyklus für alle drei maskierbaren Unterbrechungsmethoden gleich ist, beträgt die Antwortzeit für Unterbrechungen nach Methode 2 nicht mehr als für die einfachste Unterbrechungsart nach Methode 1.

2. Schritt

Zu der Software in diesem Versuch gehören drei neue Routinen, und zwar INIT2, SERV2 und SERV3. Nachstehend eine kurze Beschreibung:

INIT2: Eine Initialisierungs-Routine mit folgenden Funktionen:

1. Setzen des Interrupts auf Methode 2 durch Ausführung des Befehls IM2.
2. Einrichten der Interrupt-Vektortabelle, bestehend aus den Ausgangs-Speicherstellen für Programmunterbrechungen SERV1, SERV2 und SERV3. Tabelle 6-3 definiert die Interrupt-Vektortabelle.

Tabelle 6-3. Interrupt-Vektortabelle gemäß Routine INIT2.

Speicherstelle	Inhalt
TABLE	LO-Byte der Adresse SERV1
TABLE + 1	HI-Byte der Adresse SERV1
TABLE + 2	LO-Byte der Adresse SERV2
TABLE + 3	HI-Byte der Adresse SERV2
TABLE + 4	LO-Byte der Adresse SERV3
TABLE + 5	HI-Byte der Adresse SERV3

(LO = niedrigwertiges Byte der Adresse; HI = höchstwertiges Byte der Adresse bei einem 16-Bit Adreß-Wort.)

Anmerkung: Das I-Register ist mit der höchstwertigen Hälfte der Adreß-Tabelle geladen, die zusammen mit den Adressen von SERV1, SERV2 und SERV3 im Anhang A, Tabelle A-1 zu finden sind.

3. Das I-Register mit dem Befehl LD I,A initialisieren.

4. Maskierbare Unterbrechungen freigeben, d.h. den EI-Befehl ausführen.

SERV2: Diese Routine besteht aus nur einem Befehl: HALT.

SERV3: Diese Routine erzeugt die Anzeige, die den Inhalt von IFF1, den Zählerinhalt und die Stack-Pointer-Adresse anzeigt. Die Programmunterbrechung SERV3 erhöht das Zählwerk für ca. 10 Sekunden mit jedem Schritt um 2.

Der Steuerungsablauf zwischen INIT2, MAIN, SERV1, SERV2 und SERV3 ist in Bild 6-13 dargestellt.

LED-Monitor weiter am $\overline{\text{BHALT}}$-Signal angeschlossen lassen. Die externe Anzeige macht deutlich, wann die Z-80-CPU einen HALT-Zustand einnimmt. Die Schaltung für einen solchen LED-Monitor zeigt Bild 6-10.

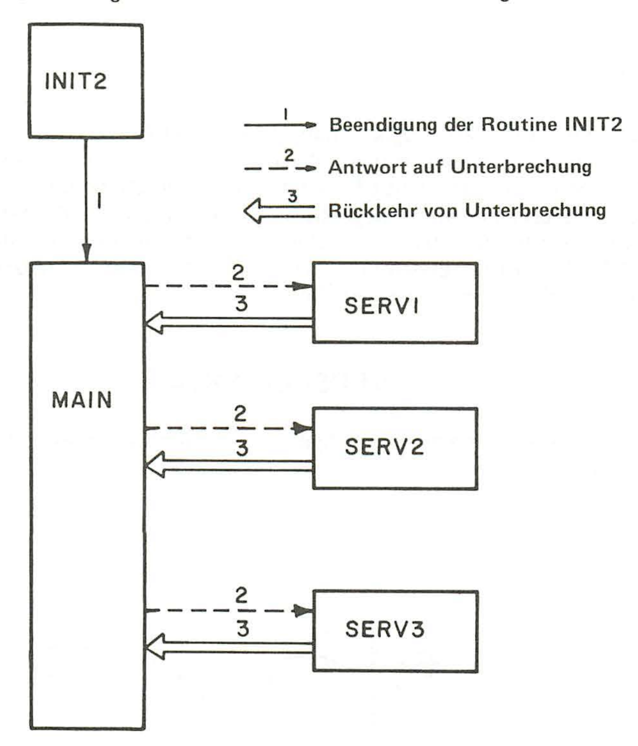


Bild 6-13. Steuerungsablauf zwischen den Routinen INIT2, MAIN, SERV1, SERV2 und SERV3.

3. Schritt

Mit den Logik-Schaltern das LO-Byte der SERV1-Adresse einstellen (6E; siehe auch Master-Symboltabelle in Anhang A). Beginnen Sie mit der Ausführung des Programms bei INIT2. Was beobachten Sie?

Der Nanocomputer® führt die Programmunterbrechung SERV1 aus, was das Zählverhalten deutlich macht. Der Zählerinhalt ändert gegenüber der MAIN-Routine mit halber Geschwindigkeit.

4. Schritt

Logik-Schalter auf das LO-Adreß-Byte (6B) der SERV3-Routine einstellen. Starten Sie das Programm erneut bei INIT2 und erzeugen einen Interrupt-Impuls. Was beobachten Sie?
Der Zählerinhalt nimmt für eine Zeitspanne von ca. 10 Sekunden mit jedem Durchlauf der Routine um 2 zu. Die Zählgeschwindigkeit ist dabei um die Hälfte geringer als bei der MAIN-Routine. Durch das Interrupt-Signal hat die Routine SERV3 die Steuerung übernommen.

Anmerkung: Im nächsten Versuch wird das Interrupt-Vorbereitungs-Flipflop benutzt.

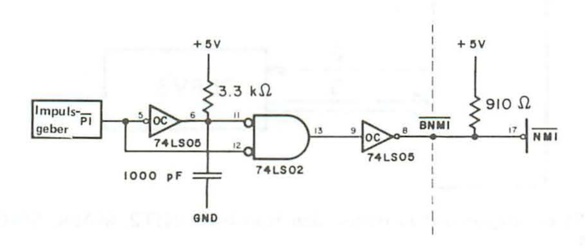
5. Schritt

Wählen Sie mit den Logik-Schaltern das LO-Adreß-Byte der SERV2 Routine vor. Was passiert mit der Anzeige, sobald das Programm bei INIT2 gestartet ist und ein Interrupt-Signal eine Unterbrechung einleitet?
Das Display erlischt; alle 7-Segment-Anzeigen sind dunkel. Dafür leuchtet der LED-Monitor für HALT auf. Das Interrupt-Signal hat die Steuerung an die SERV2-Routine übertragen. Um den HALT-Zustand zu verlassen, können Sie die RESET-Taste drücken.

VERSUCH NR. 4

Der vierte Versuch macht die Aufgabe der nichtmaskierbaren Unterbrechung deutlich.

Bild 6-14.



Für die Versuchs-Durchführung ist neben der Schaltung aus Bild 6-14 noch die Schaltung aus Bild 6-8 erforderlich.

Programme INIT1N und SERVN
Zusätzlich zu den Routinen MAIN und SERV1 benutzt dieser Versuch die folgende Software:

Objekt-Code		Quell-Code	Bemerkung
		NAME INIT1N	
3EC3	INIT1N:	LD A,0C3H	; Erstes Byte ist Sprung.
326600		LD (0066H),A	; Nichtmaskierbares Interrupt.
FD211903		LD IY,SERVN	; Serviceadresse für nicht-
FD226700		LD (0067H),IY	; maskierbare Unterbrechung.
ED56		IM1	; Interrupt-Modus 1.
3EC3		LD A,0C3H	, Erstes Byte ist Sprung.
323800		LD (0038H),A	
FD216E02		LD IY,SERV1	; Adresse der Service-
FD223900		LD (0039H),IY	; Routine 1.
08		EX AF,AF'	; Leerstellen für
3E40		LD A,40H	; CONVDI setzen.
08		EX AF,AF'	
C3C302		JP MAIN	; Sprung zur MAIN-Routine.

Objekt-Code		Quell-Code	Bemerkung
		NAME SERVN	
C5	SERVN:	PUSH BC	; Inhalte der CPU-Register
D5		PUSH DE	; zwischenspeichern.
E5		PUSH HL	
F5		PUSH AF	
DDE5		PUSH IX	
FDE5		PUSH IY	
DD23	DSN:	INC IX	; Stack-Pointer aktualisiern.
DD23		INC IX	
DD23		INC IX	
00		NOP	; keine Operation.
DD3600FF		LD (IX+00H),0FFH	; DLOOPN-Zeit setzen.
DD36010A		LD (IX+01H),00AH	; CLOOPN-Zeit setzen.
DD360202	CLOOPN:	LD (IX+02H),02H	; DLOOPN;Zeit setzen.
21E50F		LD HL,ADDH	; auf Anzeigepuffer hinweisen.
ED57		LD A,I	; Wert von IFF2 suchen.
EA4003		JP PE,HIGHN	
3600	LOWN:	LD (HL),00H	; Wert = 0
1802		JR NEXTN	
3610	HIGHN:	LD (HL),10H	; Wert = 1
ED73E20F	NEXTN:	LD (DATAL),SP	; SP-Inhalt zum Puffer über tragen.
21B90F		LD HL,LEDL	; für CONVDI setzen.
11E50F		LD DE,ADDH	; für CONVDI setzen.
CD7CFA		CALL CONVDI	
CD09F9	DLOOPN:	CALL DISPL	
DD3500		DEC (IX+00)	; Zeitgeber für Anzeige.
20F8		JR NZ,DLOOPN	
DD3502		DEC (IX+02)	; Zeitgeber für Anzeige.
20F3		JR NZ,DLOOPN	
DD3501		DEC (IX+01)	; Zeitgeber für Service- ; Routine.
20CF		JR NZ,CLOOPN	
FDE1		POP IY	; Inhalt der CPU-Register
DDE1		POP IX	; umspeichern,
F1		POP AF	

```
E1          POP HL
D1          POP DE
C1          POP BC                    ; von nichtmaskierbarem In-
ED45        RETN                      ; terrupt zurückkehren.
```

1. Schritt

Die nichtmaskierbare Unterbrechung unterscheidet sich deutlich von der in den Versuchen 1 . . . 3 beschriebenen maskierbaren Unterbrechung.

1. Für die Erzeugung eines nichtmaskierbaren Interrupts wird ein anderer Eingangspin der Z-80-CPU benutzt; er ist mit $\overline{\text{NMI}}$ bezeichnet.
2. Anders als bei den maskierbaren Unterbrechungen, bei denen die Antwort der Z-80-CPU softwaregesteuert ist (EI, DI, IM0, IM1, IM2), wird die nichtmaskierbare Unterbrechung von der CPU nie ignoriert und immer in der gleichen Weise übersetzt. Es gibt also nur einen nichtmaskierbaren Interrupt-Modus.
3. Die $\overline{\text{NMI}}$-Leitung ist eine einstufige Unterbrechung. Ein Gerät, das eine nichtmaskierbare Unterbrechung anfordert, braucht nur das $\overline{\text{NMI}}$-Signal zu aktivieren. Ein zusätzliches Erkennungs-Byte, wie bei den maskierbaren Unterbrechungen der Methoden 0 und 2, ist nicht erforderlich. Durch die NMI-Anforderung führt die CPU einen Wiederanlauf zur Speicherstelle hex 0066 durch.
4. Die Zeitfolge für Anforderung und Bestätigung eines nichtmaskierbaren Interrupts ist anders als bei einem maskierbaren. Bild 6-15 zeigt ein Zeitdiagramm für die nichtmaskierbare Interrupt-Anforderungsoperation.

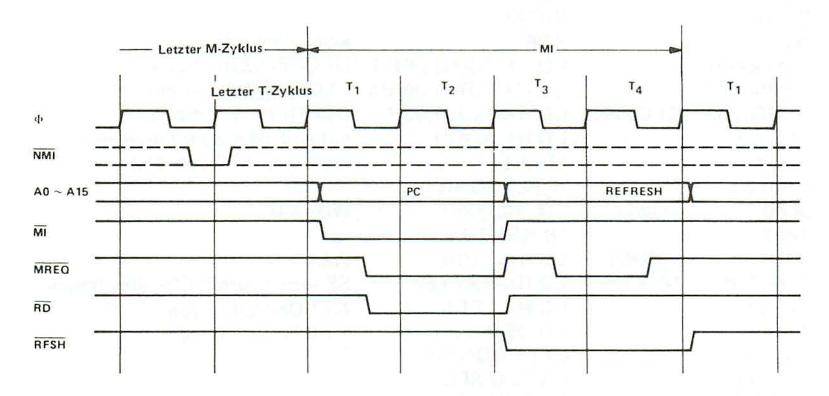


Bild 6-15. Nichtmaskierbare Interrupt-Anforderungsoperation.

5. Die CPU tastet den $\overline{\text{NMI}}$-Pin während der ansteigenden Flanke des letzten T-Zyklus im letzten M-Zyklus aller Befehle ab. Falls das $\overline{\text{NMI}}$-Signal aktiv ist (im Zustand logisch 0), läuft ein normaler Operationscode-Abrufzyklus ab, wobei die Signale $\overline{\text{M1}}$, $\overline{\text{MREQ}}$ und $\overline{\text{RD}}$ aktiviert werden. Infolgedessen findet eine Speicher-Auffrischung statt. Ein Interrupt-Bestätigungssignal (wie die NOR-Verknüpfung $\overline{\text{M1}}$ und

$\overline{IORQ}$ bei maskierbaren Unterbrechungen) erfolgt nicht, da es nicht notwendig ist. Das externe Gerät wird in jedem Fall bedient. Es besteht keine Notwendigkeit das Gerät nach einem Erkennungs-Byte abzutasten. Während des M1-Zyklus ignoriert die CPU den Inhalt des Daten-BUS und schiebt den Inhalt des PC-Registers auf den Stapelspeicher.

6. Die nichtmaskierbare Unterbrechung hat bei allen Unterbrechungen höchste Priorität. Die Bearbeitung einer maskierbaren Unterbrechung tritt immer zu Gunsten einer nichtmaskierbaren Unterbrechung zurück. Es gibt nur ein Signal mit höherer Priorität als das $\overline{NMI}$-Signal, nämlich das $\overline{BUSRQ}$-Signal. Es fordert die CPU auf, zu Gunsten einer externen CPU auf den Adreß-, Daten- und Steuer-BUS zu verzichten.
7. Hat die CPU ein nichtmaskierbares Interrupt angenommen, sind die maskierbaren Interrupts stets blockiert. Dies ist der Fall, wenn das Interrupt-Flipflop IFF1 zurückgesetzt ist. Nach Beendigung der MNI-Bearbeitung ist es wünschenswert, das Flipflop IFF1 wieder in den Zustand vor der Interrupt-Bearbeitung zurückzusetzen. Das geschieht mit Hilfe von IFF2. Das Flipflop hat vor der Interrupt-Bearbeitung den Zustand IFF1 übernommen und bleibt während der MNI-Bearbeitung unverändert. Mit dem RETN-Befehl nimmt IFF1 wieder den ursprünglichen Zustand an. Den speziellen RET-Befehl RETN (hex ED45) benutzen nur MNI-Routinen, um die CPU zu verlassen und – wie bereits erwähnt – Flipflop IFF1 wieder in den ursprünglichen Zustand zu bringen.

Man kann mit keinem Z-80-Befehl den Inhalt von Flipflop IFF1 direkt prüfen. Wohl aber können die Befehle LD A,I oder LD A,R den Zustand von IFF2 überprüfen, in dem beide Befehle den Inhalt von IFF2 ins P/V-Flag übertragen. Daher zeigt die erste Anzeige (von links) immer den Zustand von IFF2 und nicht von IFF1 an. Bei den bisherigen Versuchen 1 . . . 3 ist dieser Umstand ohne Bedeutung, da bei den maskierbaren Interrupts beide Flipflops den gleichen Zustand einnehmen. Bei nichtmaskierbaren Interrupts schließt die Bearbeitung Operationen bei IFF1 und nicht bei IFF2 ein. Deshalb werden in diesem Versuch Unterschiede deutlich.

2. Schritt

Bauen Sie die Schaltungen nach Bild 6-8 und Bild 6-14 auf. Die Literatur über den Z-80 beschreibt das NMI-Signal als ein *flankengetriggertes Signal.* Wie Sie aus der Schaltung Bild 6-14 ersehen, verzögern der Widerstand (3,3 kilo-Ohm) und der Kondensator (1000 pico-Farad) die Ankunft einer logischen 1 an Pin 11 des NOR-Gatters 74LS02. Dadurch entsteht am Ausgang des Gatters (Pin 13) ein negativer Impuls von ca. 500 ns Dauer. Die Bilder 6-15 und 6-16 zeigen die relative Zeitfolge der in dieser Schaltung interessierenden Signale. Das aktivierte $\overline{NMI}$-Signal darf nur zwischen 80 und 500 ns aktiv sein.

3. Schritt

Neben den Routinen MAIN und SERV1 finden zwei neue Routinen in diesem Versuch Verwendung.

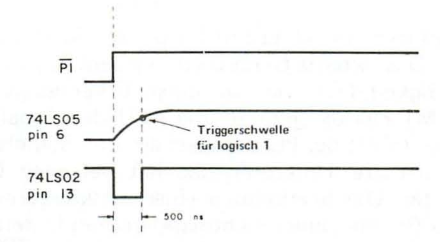


Bild 6-16. Erzeugung eines NMI-Signals von 500 ns.

INIT1N: Eine Initialisierungs-Routine mit folgenden Funktionen:
1. Laden der Adresse von SERVN in die Speicherstellen 0066 und 0067.
2. Setzen der maskierbaren Unterbrechungsmethode auf 1.
3. Laden der Adresse von SERV1 in die Speicherstellen 0038 und 0039.
4. Für Abruf von Subroutine CONVDI einrichten.

SERVN: Das ist die nichtmaskierbare Programmunterbrechung mit folgenden Funktionen:
1. Zwischenspeichern des CPU-Zustands.
2. Konstanthaltung des Zählwerks für 10 Sekunden.
3. Anzeige des Zählerinhalts.

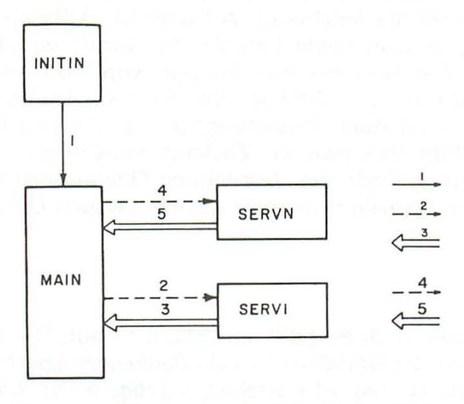


Bild 6-17. Steuerungsablauf zwischen INIT1N, MAIN, SERVN und SERV1.
1 = Beendigung der INIT1N-Routine
2 = Antwort auf maskierbare Unterbrechung
3 = Rückkehr von maskierbarer Unterbrechung
4 = Antwort auf nichtmaskierbare Unterbrechung
5 = Rückkehr von nichtmaskierbarer Unterbrechung

Laden Sie die genannten Routinen und lassen das Programm ab der Speicherstelle INIT1N ablaufen. Erzeugen Sie eine maskierbare Unterbrechung durch Schalten des Impulsgebers P0. Die Software setzt den Interrupt-Modus 1. Das veranlaßt die CPU, einen Wiederanlauf bei Speicherstelle 0038 auszuführen. Was beobachten Sie?

Der Stapelzeiger (Stack-Pointer) startet bei 0F00 und ändert nach 0EEE. Die Programmunterbrechung SERV1 wird ausgeführt.

4. Schritt

Eine nichtmaskierbare Unterbrechung kann die Z-80-CPU nicht ignorieren. Um dies zu demonstrieren, setzen Sie den Nanocomputer® zurück und ändern den ersten Befehl in der Routine MAIN von EI in NOP (hex 00). Der EI-Befehl wird nicht vor dem vorletzten Befehl in SERV1 ausgeführt! Wird ein nichtmaskierbares Interrupt bearbeitet, sind maskierbare Interrupts blockiert.
Beginnen Sie wieder mit der Programmausführung an Speicherstelle INIT1N. Die Interrupt-Flipflops IFF1 und IFF2 befinden sich beide im Zustand logisch 0. Dadurch sind maskierbare Interrupts blockiert. Versuchen Sie durch Schalten des Impulsgebers P0 eine maskierbare Unterbrechung zu erzeugen. Das ist nicht möglich. Die MAIN-Routine setzt ihre normale Ausführung fort. Es wird keine maskierbare Unterbrechung erzeugt. Die CPU ignoriert die Signale an dem $\overline{INT}$-Eingang. Schalten Sie Impulsgeber $\overline{P1}$, um eine NMI-Anforderung zu erzeugen. Was beobachten Sie?
Die Steuerung übernimmt die SERVN-Routine, die den Zählerinhalt 10 Sekunden lang konstant hält. Der Stapelzeiger wird auf 0EEE geschoben und die nichtmaskierbare Unterbrechung bearbeitet. Schlußfolgerung: Die Blockierung von maskierbaren Interrupts beeinflußt nicht die Erkennung und Bearbeitung von NMI-Signalen durch die Z-80-CPU.

5. Schritt

Geben Sie an der Speicherstelle MAIN den EI-Befehl wieder ein (00 durch FB ersetzen). Durch Schalten des Impulsgebers P0 eine maskierbare Unterbrechung erzeugen. Während die Programmunterbrechung ausgeführt wird, mit $\overline{P1}$ eine nichtmaskierbare Unterbrechung zu erzeugen. Was beobachten Sie?
Obwohl IFF1 und somit auch IFF2 zurückgesetzt sind, hat die nichtmaskierbare Unterbrechung höchste Priorität; sie wird ausgeführt. Das erkennen Sie an der stillstehenden Anzeige. Der Zählerinhalt bleibt durch die Bearbeitung der SERVN-Routine 10 Sekunden konstant. Anschließend geht die Steuerung an die SERV1-Routine zurück, die ab der unterbrochenen Stelle fertig bearbeitet wird. Die Programmsteuerung übernimmt dann die MAIN-Routine.

6. Schritt

Erzeugen Sie mit dem Impulsgeber $\overline{P1}$ ein nichtmaskierbares Interrupt. Während die Bearbeitung läuft, betätigen $\overline{P1}$ noch einmal, so daß ein zweites Interrupt-Signal zur CPU gelangt. Wie reagiert der Nanocomputer® auf das zweite Signal? Die erste Interrupt-Bearbeitung wird unterbrochen. Es beginnt sofort die Bearbeitung der zweiten Unterbrechung. Dabei geht der Stapelzeiger für 10 Sekunden auf 0EDC, springt anschließend auf 0EEE zurück. Der Nanocomputer® schließt die Bearbeitung des ersten Interrupts ab und überträgt die Steuerung wieder an die MAIN-Routine.

7. Schritt
Wenn Sie mit $\overline{P1}$ ein nichtmaskierbares Interrupt erzeugen, bleibt der angezeigte Flipflopzustand logisch 1. Wie reagiert nun der Nanocomputer® auf ein maskierbares Interrupt? Zunächst überhaupt nicht. Ist diese Reaktion richtig, obwohl für das Interrupt-Flipflop der Zustand logisch 1 angezeigt wird; das Flipflop also gesetzt ist? Ja, die Reaktion ist in Ordnung. Denken Sie daran, das nur der Zustand von IFF2 angezeigt wird und man den Zustand von IFF1 also nur indirekt über IFF2 anzeigen kann. Das Flipflop IFF2 speichert den Zustand von IFF1, bevor die CPU ein NMI-Signal empfängt. Da also keine Möglichkeit besteht, den Zustand von IFF1 direkt anzuzeigen, bleibt nur der Umweg über IFF2. In diesem Schritt hat die CPU jedoch keine maskierbaren Interrupts empfangen und somit auch nicht bestätigt. Das Flipflop IFF1 befindet sich deshalb im Zustand logisch 0, während das Display den Zustand von IFF2 anzeigt. Das bedeutet: Ein nichtmaskierbares Interrupt setzt IFF1 sofort zurück.
Das maskierbare Interrupt wird vom Interrupt-Vorbereitungs-Flipflop (Bild 6-8) gespeichert und nach Beendigung der SERVN-Routine verarbeitet.

8. Schritt
Bieten Sie der CPU ein nichtmaskierbares Interrupt-Signal an. Während der Bearbeitung dieser Unterbrechung erzeugen Sie mit P0 ein markierbares Interrupt. Wie bereits erwähnt, nimmt die CPU die zweite Unterbrechung nicht sofort an. Sie wird im Interrupt-Vorbereitungs-Flipflop gespeichert. Geben Sie noch während der Bearbeitung des ersten Interrupts ein zweites nichtmaskierbares Interrupt-Signal zur CPU und *beobachten die Bearbeitungsreihenfolge der noch zwei ausstehenden Unterbrechungen.* Sie erleben eine Überraschung.
Die CPU bearbeitet die maskierbare Unterbrechung vor der nichtmaskierbaren Unterbrechung. Für dieses Phänomen gibt es keine einleuchtende Erklärung. Es ist eine Eigenart der Z-80–CPU.

Anmerkung: Im nächsten Versuch wird die Schaltung 6-14 zur Erzeugung einer NMI-Anforderung benötigt. Der 6. Versuch arbeitet wiederum mit beiden Impulsgebern.

VERSUCH NR. 5

Der fünfte Versuch macht Sie mit dem Begriff *Kontext-Schaltung* vertraut und demonstriert dafür zwei Techniken bei der Z-80-Interrupt-Bearbeitung.

Programme
Für den Versuch sind die Routinen INIT1N, MAIN und SERVN erforderlich.

1. Schritt
Der Versuch untersucht die Bedeutung der ersten und letzten sechs Befehle in der SERVN-Routine:

```
PUSH BC
PUSH DE
PUSH HL
PUSH AF
PUSH IX
PUSH IY
●  ●  ●
POP IY
POP IX
POP AF
POP HL
POP DE
POP BC
```

Die Befehle PUSH und POP speichern die Registerinhalte der CPU um. Die Befehle halten den *Kontext* oder auch die *Umgebung* des zu unterbrechenden Programms vor Interrupt-Beginn fest. Dazu werden alle CPU-Daten des unterbrochenen Programms in einen Speicher eingelesen, wo die Programmunterbrechung sie nicht verändern kann. Nach Beendigung der Interrupt-Bearbeitung übertragen die sechs letzten Befehle wieder alle Daten in die ursprünglichen Register. Die weitere Ausführung des unterbrochenen Programms kann ordnungsgemäß erfolgen, da die CPU-Daten wieder unverändert zur Verfügung stehen und der erste Befehl nach der Unterbrechung bekannt ist. Diesen ganzen Komplex bezeichnet man als *Kontext-Schaltung*. Dabei wird also die Steuerung einer CPU von einem Vorgang der einer Funktion (das unterbrochene Programm MAIN) zu einem anderen Vorgang oder einer anderen Funktion (der Programmunterbrechung SERVN) übertragen. Die Kontext-Schaltung umschreibt also den Vorgang, welcher die unterbrochene Programmumgebung zwischenspeichert; die weitere Ausführung kann dann vom Unterbrechungspunkt wieder aufgenommen werden.
Den Kontext-Vorgang einleiten kann entweder die Interrupt-Software-Routine oder die CPU als Teil der Erkennung oder Bestätigung einer Unterbrechung. Verschiedene Computersysteme speichern den Inhalt aller CPU-Register um, bevor die Interrupt-Routine beginnt. Dies besorgt eine schnelle interne Schaltung. Der Hauptvorteil dieser Technik ist die Geschwindigkeit. Doch ist damit auch ein Nachteil verbunden. Es passiert, daß mehr Registerinhalte als notwendig umgespeichert werden und somit

mehr Speicherkapazität als notwendig. Beim Z-80 ist für die Kontext-Schaltung die Interrupt-Routine verantwortlich. Das Zwischenspeichern der CPU-Registerinhalte zum Stack-Pointer (Stapelregister) besorgen bei der SERVN-Routine die Befehle PUSH und POP. Die SERVN-Routine benutzt alle CPU-Register, so daß man die kompletten Registerinhalte zwischenspeichern muß. Benutzt eine Routine jedoch nur eine Teilmenge der CPU-Register, genügt es, den Inhalt der benutzten Register umzuspeichern. Das erspart Zeit. Die Kontext-Schaltung benötigt in der SERVN Routine 56,8 Mikrosekunden für die Ausführung. Das geht aus nachstehender Aufstellung hervor.

Befehl	**Anzahl der T-Zyklen a 0,4 μs**
PUSH AF	11
PUSH BC	11
PUSH DE	11
PUSH HL	11
PUSH IX	15
PUSH IY	15
POP AF	10
POP BC	10
POP DE	10
POP HL	10
POP IX	14
POP IY	14
	142

Bei 2,5 MHz oder ca. 400 ns pro T-Zyklus dauert es also 56,8 Mikrosekunden, bis SERV1 eine Kontext-Schaltung durchgeführt hat. Für viele Anwendungen ist das schnell genug, für andere aber ist eine schnellere Alternative erforderlich.

Die Z-80 stellt eine solche Alternative zur Verfügung, indem sie folgende Befehle benutzt:

EXX: Ausführungszeit 4 T-Zyklen
EX AF,AF': Ausführungszeit 4 T-Zyklen

Diese beiden Befehle tauschen den Inhalt der Register A, B, C, D, E, H, L und F gegen die alternativen Register A', B', C', D', E', H', L' und F' in 8 T-Zyklen oder 1,6 Mikrosekunden bei 2,5 MHz aus. Somit benötigen beide Befehle für den Kontext-Vorgang einschließlich dem Ein- und Auslesen des Stapelspeichers von IX und IY durch PUSH und POP nur 17,6 Mikrosekunden, denn $(2 \times 1{,}6) + (15 + 14) \times 2 \times 0{,}4 = 17{,}6\ \mu s$.

Bei der Anwendung der Befehle EXX und EX AF,AF' gibt es zwei bedeutende Einschränkungen: SIE DÜRFEN NICHT FÜR GESCHACHTELTE INTERRUPT-BEARBEITUNGEN BENUTZT WERDEN. AUSSERDEM SPEICHERN SIE NICHT DEN INHALT DES ALTERNATIVEN REGISTERSATZES UM. Dies wird bei der Beschreibung von geschachtelten Interrupts im nächsten Versuch deutlich. An dieser Stelle nur soviel,

daß bei geschachtelten Interrupts sich der Austausch mehr als einmal vollzieht und somit zwischengespeicherte Daten verloren gehen.
Falls noch nicht geschehen, bauen Sie jetzt die Schaltung aus Bild 6-14 auf. Laden Sie anschließend die Routinen INIT1N, MAIN und SERVN.

2. Schritt
Starten Sie das Programm bei INIT1N. Erzeugen Sie eine nichtmaskierbare Unterbrechung. Die Ausführung von MAIN wird unterbrochen, so daß SERVN die Zähleranzeige 10 Sekunden lang konstant hält. Wenn die Bearbeitung der nichtmaskierbaren Unterbrechung beendet ist, setzt MAIN das Zählen fort. Ersetzen Sie die PUSH- und POP-Befehle durch die Austauschbefehle EXX (D9) und EX AF,AF' (08) und füllen Sie NOPs (00) für unbenutzte Bytes ein. Starten Sie erneut bei INIT1N. Was passiert?
10 Sekunden lang erscheint eine irrelevante Anzeige; es folgt dann eine weitere unleserliche Anzeige. Eine neue nichtmaskierbare Unterbrechung ändert die Anzeige, hält Sie 10 Sekunden lang konstant und ändert Sie dann wieder. Scheinbar ist die grundsätzliche Zeitfolge der Unterbrechung richtig, aber die Anzeige funktioniert nicht mehr. Was ist falsch? Beantworten Sie diese Frage selbst, indem Sie sich die Bedeutung des A'-Registers in der CONVDI-Routine ansehen.

Anmerkung: Die in diesem Versuch benutzte Schaltung ist auch für den nächsten erforderlich.

VERSUCH NR. 6

Dieser Versuch befaßt sich mit den Begriffen *geschachtelte Interrupts und ablaufvariante Programme.* Alle Programmunterbrechungen in diesem Kapitel sind ablaufvariant. Was bedeutet ablaufvariant, wie wichtig ist ablaufvariantes Programm und welches sind seine Voraussetzungen?

Programme: INIT1N, MAIN, SERV1 und SERVN.

1. Schritt
Starten Sie das Programm bei INIT1N und geben einen maskierbaren Interrupt-Impuls zur CPU. Während die Bearbeitung dieser Unterbrechung läuft, erzeugen Sie mit Hilfe von $\overline{P1}$ ein nichtmaskierbares Interrupt. Die Reaktion des Nanocomputers® ist Ihnen bereits bekannt.
Die Steuerung springt zur SERV1-Routine. Zählrhythmus und Zählrichtung ändern sich. Das nichtmaskierbare Interrupt-Signal überträgt die Steuerung sofort an die SERVN-Routine, die den Zähler für 10 Sekunden konstant hält. Anschließend übernimmt wieder SERV1 die Steuerung und beendet die unterbrochene Arbeit. Sie addiert im Sekundentakt dem Zähler je 1 hinzu, bis insgesamt 10 Additionen erfolgt sind. Dann geht die Steuerung an die MAIN-Routine zurück, die den Zählerinhalt mit doppelter Zählgeschwindigkeit vermindert.
Beide Interrupts sind ineinander verschachtelt. Bild 6-19 zeigt diesen Vorgang; Service 1 ist die maskierbare, Service 2 die nichtmaskierbare Unterbrechung.

Bild 6-18. Dieser Versuch benutzt die bereits bekannten Impulsgeber.

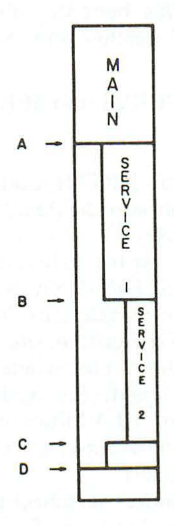


Bild 6-19. Geschachtelte Interrupts.

2. Schritt
Erzeugen Sie in schneller Folge fünf nichtmaskierbare Interrupts. Vor dem ersten Interrupt ist die Adresse des Stackpointers 0F00. Überprüfen Sie folgende Tabelle:

Anzahl der geschachtelten Interrupts	Stackpointer-Adresse
0	0EEE
1	0EDC
2	0ECA
3	0EB8
4	0EA6

Ist noch kein Interrupt erfolgt, hat der Stackpointer die Adresse 0F00. Bei einem Interrupt befindet sich der obere Teil des Stackpointers an der Speicherstelle 0EEE. Mit dem nächsten Interrupt (1. geschachtelte Unterbrechung) verschiebt sich die Adresse nach 0EDC. Bei vier Interrupts (3. geschachtelte Unterbrechung) lautet die Adresse 0EB8. Das fünfte Interrupt-Signal schiebt den Stackpointer schließlich nach 0EA6. Jede Interrupt-Bearbeitung dauert insgesamt 10 Sekunden; das entspricht der Dauer einer SERVN-Ausführung. Es wird immer nur die letzte Unterbrechung zusammenhängend ausgeführt. Die Ausführung der vorhergehenden Interrupts ist geteilt. Sie beginnt, sobald ein Interrupt-Signal eintritt. Eine weitere Unterbrechung stoppt die Bearbeitung und beginnt sofort mit dem neuen Interrupt. Erst wenn kein weiterer Interrupt erfolgt und die letzte Unterbrechung erledigt ist, kehrt der Nanocomputer® zum unterbrochenen Interrupt-Ablauf zurück und setzt dessen Bearbeitung fort. Beispiel: Es läuft die Programmunterbrechung n. Die folgende Unterbrechung n + 1 stoppt die Bearbeitung von n, so daß von den 10 Sekunden erst x Sekunden ausgeführt sind. Ist die Unterbrechung n + 1 beendet, springt der Nanocomputer® zum Interrupt n zurück; die restliche Bearbeitung dauert dort noch 10 - x Sekunden.
Bei dem geschilderten Verhalten handelt es sich immer um nichtmaskierbare Interrupts. Man kann sich jede Unterbrechung als einen separaten Vorgang vorstellen, der dasselbe Programm ausführt, nämlich SERVN. Nach zwei Unterbrechungen teilen sich zwei Vorgänge SERVN. Nach drei Unterbrechungen teilen sich drei Vorgänge SERVN usw. Nach mehreren Unterbrechungen sind mehrere Ausführungen von SERVN im Gange und alle mit einer anderen Stackpointer-Adresse. In diesem Falle ist die Folge ein geschachtelter Algorithmus, bei dem der letzte Vorgang (5. Unterbrechung) zuerst beendet wird (LIFO: Last in, first out – zuletzt hinein, zuerst heraus; das ist das charakteristische Verhalten des Stackpointers).

3. Schritt
Die I/O-Einheit (Input/Output = Eingang/Ausgang) arbeitet nach dem Interrupt-Prinzip, bei welcher der Daten-Austausch zwischen der CPU und ihren externen Geräten wie folgt abläuft:
1. Ein externes Gerät hält eine Information für die CPU bereit.

2. Der CPU wird dies durch eine Interrupt-Anforderung vom externen Gerät mitgeteilt.
3. Die Unterbrechung wird bearbeitet, indem ein Eingabebefehl mit dem unterbrechenden externen Gerät als Eingabe-Gatter ausgeführt wird.

Für die Ausgabe gilt:
1. Das externe Gerät ist bereit, ein Ausgabe-Byte anzunehmen.
2. Das externe Gerät teilt dies der CPU mit, indem es eine Interrupt-Anforderung sendet.
3. Die CPU bearbeitet die Unterbrechung, indem sie einen Ausgabebefehl als Ausgabe-Gatter ausführt.

Beispiel: Die CPU ist über ein I/O-System nach dem Interrupt-Prinzip mit mehreren gleichen Terminals verbunden. Jedes Terminal hat einen Benutzer, der in regelmäßigen Abständen ein Informations-Byte als Eingabe für die CPU eintippt. Jeder Tastendruck ist eine Unterbrechung, welche die CPU bearbeitet. Sie nimmt das Interrupt-Signal an und gibt die Information an einen speziell für das Terminal bestimmten Speicherblock weiter. Die Programmunterbrechung hat dabei mit Ausnahme für das Eingabe-Gatter und das Eingabe-Byte für alle Terminals denselben Code. Diese Situation kann die Software durch zwei alternative Techniken lösen.

Technik Nr. 1: *Mehrere Kopien*
Bei dieser Technik ist für jedes Terminal eine Interrupt-Service-Routine abgespeichert. Jedes Terminal liefert bei einem angeforderten Interrupt einen Erkennungscode. Aufgrund dieses Erkennungscodes geht die Steuerung an die richtige Kopie der Interrupt-Service-Routine über. Die einzelnen Kopien unterscheiden sich nur durch zwei Bytes, das ist einmal der Gerätecode im IN-Befehl und zum anderen das niederwertige Byte der Speicheradresse für das Eingabezeichen.
Technik Nr. 2: *Gemeinsamer Code*
Alle Terminals benutzen dieselbe Programmunterbrechung. Dies geschieht in ähnlicher Weise wie im 2. Schritt dieses Versuchs, wo dieselbe Unterbrechung in sich selbst geschachtelt ist. Diese Art der Programmunterbrechung muß mehrere wichtige Funktionen einschließen, die bei Technik Nr. 1 nicht unbedingt notwendig sind:
a) Das erkannte Terminal ist über eine Logik mit dem I/O-Gatter und der vom Eingangscode definierten Speicherstelle zu verbinden.
b) Eine weitere Logik muß dafür sorgen, daß man die Programmunterbrechung durch sich selbst unterbrechen kann. Außerdem müssen mehrere Interrupts gleichzeitig aktiv sein können, ohne daß sich die Registerinhalte verändern. Mit anderen Worten: Die Programmunterbrechung muß *ablaufvariant* sein.

Folgende Bedingungen gewährleisten ein ablaufvariantes Programm:
1. Der Code bleibt unverändert.
2. Die Inhalte der CPU-Register werden beim Eintritt in die Routine gerettet, d.h. zwischengespeichert und bei Beendigung der Routine an die CPU zurückgegeben.
3. Die mit jedem aktiven Eintritt in die Routine verbundenen Datenräume schließen sich aus, d.h. dürfen sich gegenseitig nicht beeinflussen.

Alle in diesen Versuchen benutzten Programme sind ablaufvariant. Der nächste Schritt macht anhand der SERVN-Routine dies noch deutlicher.

4. Schritt
Prüfen Sie die SERVN-Routine, um festzustellen, ob die genannten Bedingungen wirklich ausreichen. In der Tat, das Programm der SERVN-Routine ändert sich nicht. Der Inhalt der CPU-Register wird durch die Befehle PUSH und POP zu Beginn und am Schluß der Routine beibehalten. Das IX-Register enthält die Speicher-Hinweisadresse. Bei den drei Speicherstellen IX + 00, IX + 01 und IX + 2 handelt es sich um Zeit-Bytes. Mit ihrer Hilfe hält die SERVN-Routine die Anzeige ca. 10 Sekunden konstant. Wird die SERVN-Routine durch sich selbst oder eine andere, den Datenraum bedrohende Routine unterbrochen, ist es wichtig, die Werte dieser drei Speicherstellen festzuhalten. Nur dann ist die Anzeige für insgesamt 10 Sekunden konstant, wenn SERVN die Ausführung wieder aufnimmt. Ändert eine Interrupt-Routine diese Speicherstelle, tritt auch eine Änderung der Anzeigezeit ein. Abhängig von den geänderten Werten der Speicherstellen IX + 00, IX + 01 und IX + 02 kann die Zeit der konstant gehaltenen Anzeige sehr kurz oder unendlich lang sein.
Da es sich hierbei um Speicherstellen handelt, ist die im vorigen Versuch beschriebene Kontext-Schaltung nicht geeignet. Eine Möglichkeit, die Zeit-Bytes unverändert zu halten, ist die Zwischenspeicherung im Stackpointer. Dazu werden sie zunächst in die CPU geladen und dann der geeignete PUSH-Befehl erteilt. Bevor die Steuerung an die SERVN-Routine zurückgeht, muß erst ein POP- und dann ein Rückladebefehl erfolgen. Diese Technik ist für eine kleine Anzahl von Daten-Bytes akzeptabel. Sie ist aber zeitraubend, sobald die Zahl der Speicher-Bytes zunimmt. Um den Inhalt von zwei aufeinanderfolgenden Speicherstellen im Stackpointer zwischenzuspeichern, sind vier Befehle erforderlich:

LD HL, (MEMLOC)	16 T-Zyklen
PUSH HL	11 T-Zyklen
POP HL	10 T-Zyklen
LD (MEMLOC), HL	16 T-Zyklen
	53 T-Zyklen = 21,2 ms bei 2,5 MHz.

Es gibt zu der gerade beschriebenen Methode eine Alternative, den Datenraum in den Stackpointer (Stapelspeicher) zu schieben. Es ist die Zuweisung neuen Datenraums für das Interrupt-Programm durch Manipulation von Speicher-Hinweisadressen, die im wesentlichen auf den Anfang eines Datenraums hinweisen. Diese Methode benutzt die SERVN-Routine, in der das IX-Register die Speicher-Hinweisadresse enthält. Beachten Sie, wie die drei Befehle INC IX am Anfang der SERVN-Routine den Datenraum im Speicher drei Bytes weiter nach oben schiebt.
Die SERVN-Routine erfüllt also alle Voraussetzungen für ein ablaufvariantes Programm. Den Beweis tritt der 2. Schritt an. Bereits dort kann man erkennen, wie mühelos SERVN arbeitet und sich selbst ohne Komplikationen unterbricht.

Jedes nichtmaskierbare Interrupt hat ein Problem. Dabei spielt es keine Rolle, welche Routine die Steuerung übernimmt. Jedes nichtmaskierbare Interrupt kann ein vorhergehendes nichtmaskierbares Interrupt unterbrechen. Im Gegensatz zu den maskierbaren Unterbrechungen kann man die nichtmaskierbaren Unterbrechungen nicht blockieren. Folgen zwei nichtmaskierbare Unterbrechungen schnell genug aufeinander, ist der Bearbeitungsablauf bekannt (siehe Bild 6-19). In solchen Fällen muß man den Datenraum schützen, oder es treten unvorhersehbare Ergebnisse ein. Was passiert, wenn man den Datenraum nicht schützt?

Nehmen Sie an der SERVN-Routine folgende Änderung vor:

Speicherstelle	alter Befehl	neuer Befehl
DSN	DD	00
DSN + 1	23	00
DSN + 2	DD	00
DSN + 3	23	00
DSN + 4	DD	00
DSN + 5	23	00

Durch die Änderung entfallen drei Befehle, die den Datenraum für die SERVN-Routine mit jeder Eingabe schützen. Die Folgen werden im 5. Schritt deutlich.

5. Schritt

Starten Sie das Programm bei INIT1N und erzeugen unmittelbar hintereinander zwei nichtmaskierbare Unterbrechungen. Wielange bleibt die Anzeige konstant?

Die Bearbeitung der zweiten Unterbrechung erfolgt in der vom Programm festgelegten Zeit (ca. 10 Sekunden). Die erste Unterbrechung dauert jedoch ca. 2 Minuten!

6. Schritt

Prüfen Sie, ob es sich bei der Interrupt-Service-Routine SERV1 um ein ablaufvariantes Programm handelt. Solange SERV1 die Steuerung hat, sind maskierbare Interrupts blockiert. Um zu prüfen, ob es sich um ein ablaufvariantes Programm handelt, muß man dies ändern. Dazu tauschen Sie den NOP-Befehl an der Speicherstelle DS1 + 6 gegen den EI-Befehl aus. Dieser Befehl speichert die CPU-Registerinhalte um und aktualisiert den Datenraumzeiger (Stackpointer). Der Hex-Code für den Befehl EI ist FB. Beginnen Sie mit der Programmausführung an der Speicherstelle INIT1. Überprüfen Sie folgende Tabelle:

Interrupt Nr.	Stackpointer
0	0F00
1	0EEE
2	0EDC
3	0ECA
4	0EB8
5	0EA6

Fazit: Die Unterbrechungen sind geschachtelt; die Bearbeitung erfolgt ohne Komplikation. Es handelt sich also um ein ablaufvariantes Programm.

Kapitel 7

Z-80 Parallel I/O (PIO)

Die Z-80 Parallel I/O (PIO) gehört zu einer IC-Gruppe, die das Interfacing mit der Z-80-CPU erleichert. Die PIO-Schaltung soll eine programmierbare TTL-kompatible parallele Datenübertragung mit zwei Gattern zwischen der Z-80-CPU und den externen Geräten gewährleisten. (In der internationalen Literatur sind die zum PIO-IC gehörenden Gatter mit dem englischen Fachausdruck "Port" bezeichnet. Auch dieses Buch hält sich in den folgenden Kapiteln an diese Bezeichnung). In einem DIP-Gehäuse (DIP = Dual-In-Line-Package) mit 40 Anschlußstiften bietet das PIO-IC mit nur wenigen externen Schaltungen folgende Leistungen:

- zwei unabhängige, parallele bidirektionale, externe 8-Bit-Interface-Ports mit Quittungsbetrieb;
- Interruptbetriebene I/O;
- vier wählbare Software Betriebsarten:
  Methode 0 - Byte-Ausgabe
  Methode 1 - Byte-Eingabe
  Methode 2 - bidirektionales Byte (nur Port A)
  Methode 3 - Bit-Steuerung
  Jede Betriebsart verwendet den interruptgesteuerten Quittungsbetrieb.
- Verkettete Interrupt-Logik;
  Alle Ein- und Ausgänge sind TTL-kompatibel.

Nach der Lektüre dieses Kapitel können Sie

- alle erwähnten PIO-Eigenschaften verstehen;
- alle erwähnten PIO-Eigenschaften untersuchen.

DAS PIO-IC

Der folgende Abschnitt macht Sie mit der Pin-Belegung beim PIO-IC vertraut. Es folgt außerdem eine Kurzbeschreibung der Betriebsarten und der Programmierung. Damit Sie möglichst schnell mit den praktischen Versuchen beginnen können, ist der theoretische Teil relativ kurz gehalten.

Pin-Beschreibung des PIO-ICs (Bild 7-1)

Bild 7-1 zeigt die Pin-Belegung beim PIO-IC.

D7 . . . D0 – Z-80-CPU-Daten-BUS (bidirektionaler Daten BUS)
Diese BUS-Leitungen übertragen alle Befehle zwischen der Z-80-CPU und dem PIO-IC. D0 ist das niederwertigste Bit. Auswahl Port A oder B, Pin 6 (Eingang, high-aktiv). Das

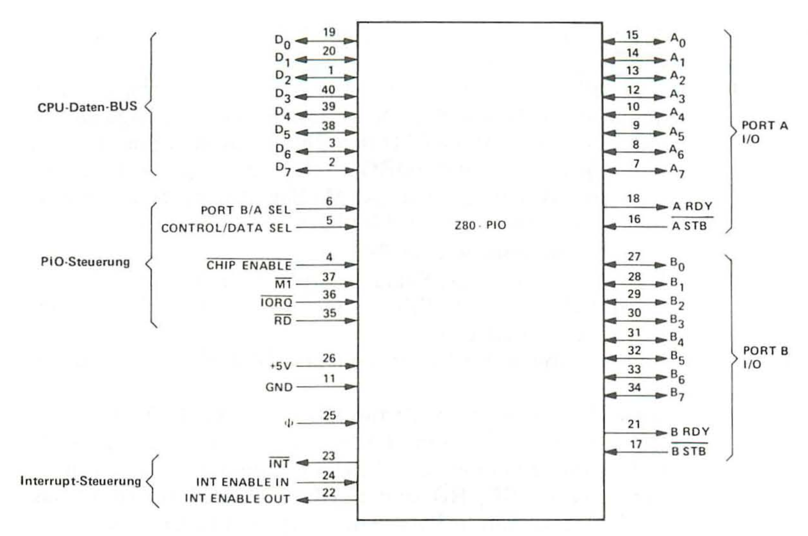


Bild 7-1. Pin-Belegung des PIO-ICs.

Signal an diesem Pin bestimmt, ob zwischen der CPU und dem PIO-IC Port A oder Port B benutzt wird. Port A ist in Funktion, wenn das Eingangssignal logisch 0 ist. Bei einem hohen Eingangspegel (logisch 1) ist Port B angewählt. Diese Auswahlfunktion übernimmt in der Regel das Adreß-Bit AO von der CPU.

Steuerungs- oder Datenauswahl, Pin 5 (Eingang, high-aktiv)
Der Spannungspegel an diesem Pin bestimmt die Art des Datentransfers zwischen der CPU und dem PIO-IC. Steht an diesem Pin während der CPU-Schreiboperation hohes Potential an, wird die Information auf dem Z-80-Daten-BUS als *Befehl* für den von der Auswahlleitung bestimmten Port übersetzt. Ist das Eingangssignal an diesem Pin logisch 0, findet eine Datenübertragung zwischen der CPU und PIO-IC statt. Für diese Auswahlfunktion steht in der Regel das Adreß-Bit A1 zur Verfügung.

$\overline{CE}$ – Chip Enable (Freigabe), Pin 4 (Eingang, low-aktiv)
Ein niedriger Pegel an diesem Pin ermöglicht dem PIO-IC Befehls- oder Daten-Eingaben bei einem Schreibzyklus von der CPU anzunehmen oder Daten zur CPU zu übertragen. Das $\overline{CE}$-Signal setzt sich meist aus vier I/O-Portzahlen zusammen. Es sind dies Port A und B sowie das Steuer- und Daten-Port.

Φ – Systemtakt, Pin 25 (Eingang)
Um bestimmte Signale intern zu synchronisieren, benutzt das PIO-IC den Z-80-Systemtakt. Es handelt sich dabei um einen Einphasentakt. Es handelt sich dabei um einen Einphasentakt.

$\overline{M1}$ Maschinenzyklus – Ein Signal von der CPU, Pin 37 (Eingang, low-aktiv)
Das CPU-Signal dient als Synchronisations-Impuls zur Steuerung mehrerer PIO-Operationen. Sind die Signale $\overline{M1}$ und $\overline{RD}$ aktiv, ruft die CPU den Befehl aus dem Speicher ab. Ist hingegen $\overline{M1}$ und $\overline{IORQ}$ aktiv, bestätigt die CPU ein Interrupt. Außerdem hat das $\overline{M1}$-Signal noch zwei weitere Funktionen innerhalb des PIO-ICs:

1) $\overline{M1}$ synchronisiert die PIO-Interrupt-Logik.
2) Gelangt ein $\overline{M1}$-Signal ohne aktiviertes $\overline{RD}$- oder $\overline{IORQ}$-Signal zur CPU, leitet dies für das PIO-IC einen Reset-Vorgang ein.

$\overline{IORQ}$ – Ein-/Ausgabe-Anforderung von der CPU, Pin 36 (Eingang, low-aktiv)
Dieses Signal wird in Verbindung mit den B/A-Auswahl-, C/D-Auswahl-, $\overline{CE}$- und $\overline{RD}$-Signalen benutzt, um Befehle und Daten zwischen der CPU und dem PIO-IC zu übertragen. Wenn $\overline{CE}$, $\overline{RD}$ und $\overline{IORQ}$ aktiv sind, überträgt das von B/A angewählte Port Daten zur CPU (eine Lese-Operation). Sind die Signale $\overline{CE}$ und $\overline{IORQ}$ aktiviert, schreibt die CPU Informationen in das von B/A adressierte Port ein. Abhängig vom anliegenden C/D-Auswahlsignal sind es entweder Daten oder Steuerungs-Informationen. Bei gleichzeitig aktiviertem $\overline{IORQ}$-Signal bestätigt die CPU ein Interrupt. Dabei legt das unterbrechende Port automatisch den Interrupt-Vektor auf den Daten-BUS der CPU, falls die Interrupt-Anforderung von einem Gerät mit höchster Priorität kommt.

$\overline{RD}$ – Zustand des Lesezyklus von der Z-80-CPU, Pin 35 (Eingang, low-aktiv)
Ein aktiviertes $\overline{RD}$-Signal löst eine MEMORY READ- oder I/O READ-Operation aus. Das $\overline{RD}$-Signal wird mit den B/A-Auswahl-, C/D-Auswahl-, $\overline{CE}$- und $\overline{IORQ}$-Signalen benutzt, um Daten von dem PIO-IC zur CPU zu übertragen.

IEI – Interrupt Enable In, Interrupt-Freigabe IN. (Pin 35 (Eingang, high-aktiv)
Dieses Signal setzt Prioritäten bei verketteten Interrupts, wenn mehr als ein interruptbetriebenes Gerät eingesetzt ist. Ist das Signal an Pin 23 logisch 1, folgert das PIO-IC, daß keine anderen Geräte höherer Priorität von der Programmunterbrechung der CPU benutzt werden.

IEO – Interrupt Enable OUT, Interrupt-Freigabe OUT, Pin 22 (Ausgang, high-aktiv)
Das IEO-Signal ist das andere Signal, mit dem eine Prioritäts-Verkettung gebildet werden kann. Es ist nur dann im Zustand logisch 1, wenn das Signal an IEI high ist und die CPU vom PIO-IC keine Unterbrechung bearbeitet. Das Signal blockiert Geräte mit niedrigerer Priorität, wenn während der Interrupt-Service-Routine die CPU ein Gerät mit höherer Priorität bedient.

$\overline{INT}$ – Interrupt-Anforderung, Pin 23 (offener Kollektorausgang, low-aktiv)
Das PIO-IC gibt zur CPU eine Interrupt-Anforderung ab, wenn das $\overline{INT}$-Signal aktiviert ist.

A0 . . . A7 – Daten-BUS für Port A (bidirektional, Tristate)
Dieser 8-Bit-Daten-BUS überträgt die Daten oder Steuerungs-Informationen zwischen Port A des PIO-ICs und einem externen Gerät. A0 ist das niederwertigste Bit.

$\overline{ASTB}$ – Abtastimpuls von externem Gerät für das Ausgabe-Register, Pin 18 (Eingang, low aktiv)
Die Bedeutung dieses Signals hängt von der für z.B. Port A gewählten Betriebsart ab:

Methode 0 – Byte-Ausgabe: Die positive Flanke dieses Abtastimpulses, der von einem externen Gerät ausgeht, bestätigt dem PIO-IC den Datenempfang.

Methode 1 – Byte-Eingabe: Der Abtastimpuls geht von einem externen Gerät aus, um dessem Daten in das Eingabe-Register von Port A zu laden. Das PIO-IC liest Daten ein, wenn das Signal aktiv ist.

Methode 3 – Bit-Steuerung: Der Abtastimpuls wird intern blockiert.

ARDY – Bereitschaft Ausgabe-Register, Pin 18 (Ausgang, high-aktiv)
Auch die Bedeutung dieses Signals hängt von der für z.B. Port A gewählten Betriebsart ab:

Methode 0 – Das Signal wird aktiv und zeigt an, daß Ausgabe-Register von Port A geladen und der Daten-BUS für die Übertragung an das externe Gerät bereit ist.

Methode 1 – Dieses Signal ist aktiviert, wenn das Eingabe-Register von Port A leer und bereit ist, Daten vom externen Gerät anzunehmen.

Methode 2 – Stehen im Ausgabe-Register von Port A Daten zur Übertragung an das externe Gerät zur Verfügung, ist das Signal aktiviert. Hierbei gelangen keine Daten auf den Daten-BUS von Port A, ehe nicht $\overline{ASTB}$ aktiv ist.

Methode 3 – Dieses Signal wird blockiert und in den Low-Zustand gezwungen.

B0 . . . B7 – Daten-BUS für Port B (bidirektional, Tristate)
Mit diesem 8-Bit-BUS werden Daten oder Steuerungs-Informationen zwischen Port B des PIO-ICs und einem externen Gerät übertragen. Der Daten-BUS von Port B kann zum Betreiben von Darlington-Transistoren 1,5 mA bei 1,5 V liefern. B0 ist das niederwertigste Bit.

$\overline{BSTB}$ – Abtastimpuls von externem Gerät für Eingabe-Register, Pin 17 (Eingang, low-aktiv)

Die Bedeutung dieses Signals ähnelt der von $\overline{ASTB}$ mit folgender Ausnahme:
Bei der bidirektionalen Methode von z.B. Port A tastet dieses Signal Daten vom externen Gerät in das Eingabe-Register von Port A.

BRDY – Bereitschaft Eingabe-Register, Pin 21 (Ausgang, high-aktiv)
Das BRDY-Signal unterscheidet sich in folgendem Punkt von ARDY:
Bei der bidirektionelen Methode von z.B. Port A ist das Signal high, wenn das Eingabe-Register von Port A leer und bereit ist, Daten vom externen Gerät aufzunehmen.

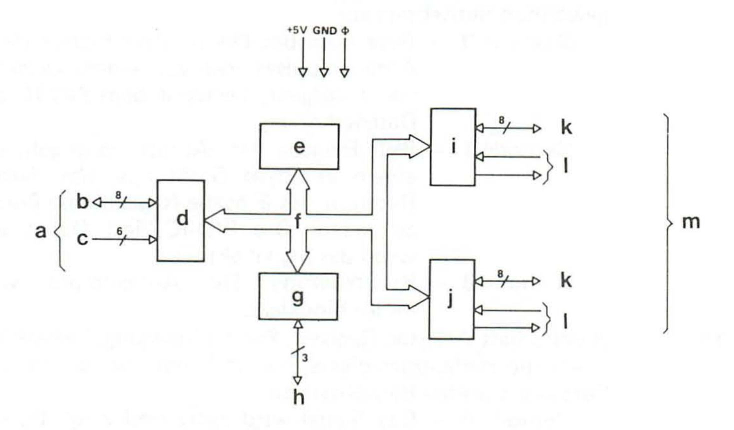


Bild 7-2. Funktionelles Blockschaltbild.
a) CPU-Interface, b) Daten-BUS, c) PIO-Steuerungen, d) CPU-I/O-BUS, e) interne Steuerlogik, f) interne-BUS-Leitung, g) Interrupt-Steuerung, h) Interrupt-Steuerleitungen, i) I/O-Port A, j) I/O-Port B, k) Daten oder Steuerung, l) Quittungsbetrieb, m) externes Interface.

Funktionsübersicht

Bild 7-2 zeigt das Blockschaltbild des PIO-ICs. Das CPU-Interface besteht aus den acht Daten-BUS und den folgenden Steuer-Leitungen: $\overline{M1}$, $\overline{IORQ}$, $\overline{RD}$, B/A Sel, C/D Sel und $\overline{CE}$. Der Daten-BUS enthält alle Daten- und Befehls-Bytes zwischen der CPU und dem PIO-IC. Die Steuerleitungen aktivieren das PIO–IC ($\overline{CE}$), bestimmen die Art des zu übertragenden Bytes (C/D), wählen Port A oder B aus (B/A) und legen die Richtung des Datenflusses fest ($\overline{M1}$, $\overline{IORQ}$ und $\overline{RD}$).
Die Funktion der CPU-I/O arbeitet mit den anderen Funktionen über den internen Daten-BUS des PIO-ICs zusammen. Die Interrupt-Steuerfunktion bedient die drei Interrupt-Steuerleitungen EI, EO und $\overline{INT}$. Die interne Steuerungslogik sorgt für die Gesamtfunktions-Synchronisation und Koordination. Die Funktionen der Ports A und B verbindet das PIO-IC mit den externen Geräten. Die acht Daten- oder Steuerleitungen sind:
für Port A: PA0, PA1, PA2, PA3, PA4, PA5, PA6 und PA7
für Port B: PB0, PB1, PB2, PB3, PB4, PB5, PB6 und PB7

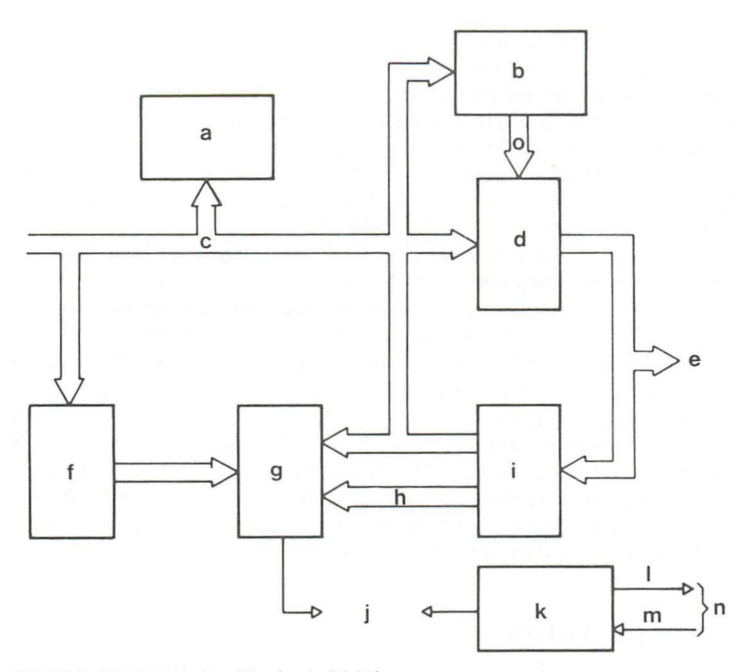


Bild 7-3. Funktionelles Blockschaltbild.
a) Methoden-Steuer-Register (2 Bits), b) I/O-Auswahl-Register (8 Bits), c) interner BUS, d) Daten-Ausgabe-Register (8 Bits), e) externer Daten- oder Steuer-BUS mit 8 Bits, f) Masken-Steuer-Register (2 Bits), g) Masken-Register (8 Bits), h) Eingabedaten, i) Daten-Eingabe-Register (8 Bits), j) Anforderungen, k) Steuerlogik für Quittungsbetrieb, l) Bereitschaft, m) Abtastimpuls, n) Leitungen für Quittungsbetrieb, o) Ausgabe-Freigabe.

Die Leitungen für den Quittungsbetrieb sind:
für Port A: $\overline{\text{ASTB}}$ und ARDY
für Port B: $\overline{\text{BSTB}}$ und BRDY.

Jede Port-I/O-Funktion hat ihr eigenes funktionelles Blockschaltbild, wie für Port A und B in Bild 7-3 dargestellt.

Das Methoden-Steuer-Register enthält den 2-Bitcode, der die laufende Betriebsart des PIO-ICs vorschreibt. Das Register wird unter Programmsteuerung über eine Befehlsbyte-Ausgabe in das PIO-IC geladen. Den genauen Vorgang zum Setzen der PIO-Betriebsart beschreibt der nächste Abschnitt. Alle vom PIO-IC zu einem externen Gerät übertragenen Daten müssen das 8-Bit-Daten-Ausgabe-Register (d) passieren. Alle vom PIO-IC aus einem externen Gerät empfangenen Daten müssen das 8-Bit-Daten-Eingabe-Register (i) passieren. Die Quittungsbetriebsleitungen (n) steuern die Datenübertragung, indem sie anzeigen, wann Daten zur Ausgabe oder zum Empfang bereit sind.

Das I/O-Auswahl-Register (b), das Masken-Steuer-Register (f) und das Masken-Register (g) werden nur in PIO-Operationen nach Methode 3 benutzt. Sie hat folgende Eigenschaften:

1. Jedes Bit der PIO-Ports kann man als Eingabe- oder Ausgabe-Bit benutzen. Das I/O-Auswahl-Register lädt unter Programmsteuerung Bit-Logikstufen mit folgender Bedeutung:
   Logisch 1 bedeutet, das Bit ist eine Eingabeleitung.
   Logisch 0 bedeutet, das Bit ist eine Ausgabeleitung.
2. Das PIO-IC kontrolliert seine Eingabeleitungen auf ein festgelegtes Bit-Muster. Wenn das Muster erscheint, erzeugt das PIO-IC automatisch eine Interrupt-Anforderung für die Z-80-CPU. Das Bit-Muster ist durch das Masken-Steuer- und das Masken-Register festgelegt. Das Masken-Register bestimmt, welcher Teil der Eingangsstifte kontrolliert und welcher maskierbar (ignoriert) werden soll. Das Masken-Steuer-Register besteht nur aus 2 Bits. Das erste Bit bestimmt, ob ein Pin im aktiven Zustand bei logisch 0 oder logisch 1 zu berücksichtigen ist. Das zweite Bit bestimmt, ob die AND- oder OR-Funktion für die unmaskierbaren Eingabeleitungen zur Erzeugung eines Interrupts relevant sind. Ist ein Masken-Steuer-Register mit den Bits 0 und 1 geladen, kann nur dann ein Interrupt erfolgen wenn sich ALLE unmaskierbaren Eingabeleitungen im Zustand logisch 0 befinden. Andererseits bestimmt 00, nur dann eine Unterbrechung zu erzeugen, wenn sich EINE unmaskierbare Eingabeleitung im Zustand logisch 0 befindet.

Programmierung des PIO-ICs

Die Programmierung des PIO-ICs unterteilt sich in drei wichtige Kategorien von PIO-Betriebs-Parametern:

1. Der Interrupt-Vektor: Das PIO unterstützt die Interrupt-Bearbeitung durch die CPU bei der Interrupt-Methode 2. Wenn also das PIO-IC die CPU unterbricht, erwartet die CPU einen Geräte-Erkennungscode (siehe Kapitel 6). Der Erkennungscode wird als niedrigwertigstes Byte einer Vektor-Tabellenadresse übersetzt, was wiederum auf den Beginn einer Programmunterbrechung hinweist. Der Erkennungscode gelangt in das PIO-IC, indem man ihn in das Steuergatter des gewünschten PIO-Ports schreibt. Das niedrigwertigste Bit des Erkennungscodes muß 0 sein, um dem PIO-IC anzuzeigen: das Steuer-Byte ist wirklich ein Interrupt-Vektor.
2. Die Betriebsart: Ein PIO-Port kann auf vierfache Weise betrieben werden: Methode 0 (Byte-Ausgabe), Methode 1 (Byte-Eingabe), Methode 2 (Byte bidirektional) oder Methode 3 (Steuerung). Die Festlegung der Betriebsart erfolgt durch die Ausgabe eines Bytes zum Steuergatter des gewünschten Ports nach folgendem Schema:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

wobei M1 und M0 folgende Bedeutung haben:
      00 bedeutet Ausgabemethode (Methode 0)
      01 bedeutet Eingabemethode (Methode 1)
      10 bedeutet bidirektionale Methode (Methode 2)
      11 bedeutet Steuermethode (Methode 3)

   Die "1"-Bits im unteren Halbbyte (Bits D0 bis D3) signalisieren dem PIO-IC: das Steuer-Byte legt die Betriebsart fest. Ist für ein gegebenes

Port die Methode 3 gewählt, übersetzt die PIO-Steuerlogik das folgende Steuer-Byte für dieses Port als I/O-Ausgabe-Wahl-Byte. Somit sind die Ein- und Ausgangsleitungen festgelegt.

3. Das Interrupt-Steuerwort. Es hat folgendes Schema:

D7	D6	D5	D4	D3	D2	D1	D0
Freigabe Interrupt	AND/ OR	High/ Low	Maskierung folgt	0	1	1	1
	nur in Methode 3 benutzt						

Was bedeuten die einzelnen Bits?

D7 – Interrupt-Freigabe: Dieses Bit setzt das Interrupt-Freigabe-Flipflop des PIO-ICs. Befindet sich das Bit im Zustand logisch 1, können alle Interrupts vom externen Gerät zur CPU das PIO-IC passieren. Befindet sich das Bit im Zustand logisch 0, werden alle Interrupt-Anforderungen vom externen Gerät ignoriert. Dieses Bit ist völlig unabhängig von Z-80-IFF1 und -IFF2.

D6 - AND/OR: Das Bit ist nur beim Betrieb nach Methode 3 von Bedeutung und bestimmt das Kriterium zur Erzeugung eines Interrupts. Logisch 1 bedeutet, alle unmaskierbaren Bits müssen zur Erzeugung einer Unterbrechung aktiv sein. Es wird also eine AND-Funktion ausgeführt. Logisch 0 bestimmt, das Vorhandensein eines aktiven unmaskierbaren Bits genügt für eine zu erzeugende Unterbrechung. Die Realisierung übernimmt die OR-Funktion.

D5 – High/Low: Dieses Bit ist ebenfalls nur beim Betrieb nach Methode 3 relevant. Es bestimmt, welcher logische Zustand aktiv sein soll, logisch 0 oder logisch 1.

D4 - Maskierung folgt: Befindet sich dieses Bit im Zustand logisch 1 und ist als Betriebsart die Methode 3 gewählt, dann ist das nächste Steuer-Byte vom PIO-IC als das Maskenbyte zu übersetzen. Dadurch findet eine Überprüfung aller Maskenbits statt. Ist das untersuchte Bit 1, hat es keine Bedeutung. Bei logisch 0 allerdings wird es zum Zwecke der Interrupt-Erzeugung kontrolliert (AND oder OR).
Falls es sich bei der Betriebsart nicht um Methode 3 handelt, veranlaßt D4 im Zustand logisch 1 eine Rücksetzung der ausstehenden Unterbrechungen, das bedeutet Anullierung.

D3 - D0: Sind diese vier Bits im Zustand 0111, übersetzt das PIO-IC das Steuer-Byte als Interrupt-Steuerwort.

In den Versuchen sammeln Sie mit fast allen der beschriebenen Programmalternativen für das PIO-IC Erfahrung. Darum wird auf eine weitere Beschreibung an dieser Stelle verzichtet.

Rücksetzen des PIO-ICs

Genau wie die Z-80-CPU, nimmt auch das PIO-IC einen definierten Anfangszustand ein, wenn man die Stromversorgung einschaltet. In diesem Fall ist das $\overline{M1}$-Signal logisch 0, ohne daß gleichzeitig $\overline{RD}$ oder $\overline{IORQ}$ aktiv sind. Während sich der Anfangszustand einstellt, geht das $\overline{M1}$-Signal auf logisch 1. Der Anfangszustand ist wie folgt definiert:

1. Beide Masken-Register auf 0 zurückgesetzt.
2. Alle Port-Datenleitungen sind in den hochohmigen Zustand versetzt.

3. Die RDY-Leitungen für den Quittungsbetrieb sind inaktiviert (low).
4. Die Betriebsart 1 ist gewählt.
5. Beide Port-Interrupt-Freigabeflipflops sind zurückgesetzt.
6. Beide Port-Ausgangsregister sind auf 0 zurückgesetzt.
7. Die Interrupt-Vektor-Adressenregister werden *nicht* beeinflußt.

Der Anfangszustand bleibt erhalten, bis das PIO-IC von der CPU ein Steuerwort erhält.

EINFÜHRUNG IN DIE VERSUCHE

Auf der PC-Platine des Nanocomputers® befinden sich zwei PIO-ICs, wie in Bild 7-4 dargestellt.

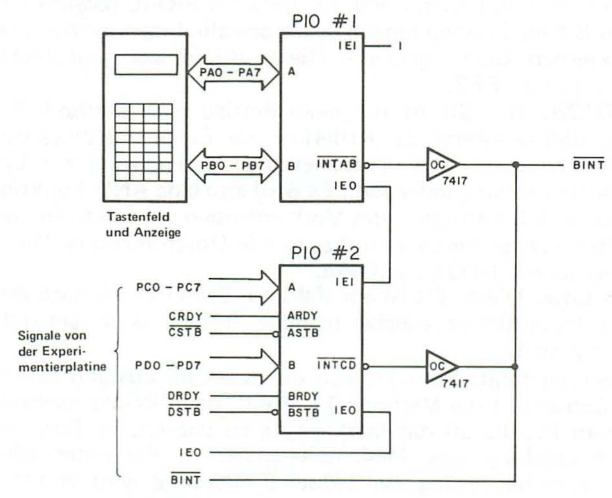


Bild 7-4. PIO-ICs des Nanocomputers®.

PIO-IC1 stellt die Verbindung mit dem Tastenfeld bzw. der Anzeige des Nanocomputers® sowie mit dem Kasetten- und Fernschreib-Interface her (siehe Kapitel 5). PIO-IC2 wird vom Betriebssystem des Nanocomputers® nicht benutzt und steht externen Schaltungen zur Verfügung. Ein Kabelstrang verbindet PIO2 über Steckerleiste J7 mit dem Anschluß C. Die Anschlüsse des ICs PIO2 stehen dem Anwender auf der Experimentierplatine am Anschluß C zur freien Verfügung. Es sind dies die Signale:

Datenleitungen für Port A: PC0, PC1, PC2, PC3, PC4, PC5, PC6 und PC7;
Quittungsbetrieb für Port A: $\overline{\text{CSTB}}$, CRDY;
Datenleitungen für Port B: PD0, PD1, PD2, PD3, PD4, PD5, PD6 und PD7;
Quittungsbetrieb für Port B: $\overline{\text{DSTB}}$, DRDY.

Die Leitungen für Port A sind mit dem Buchstaben "C" bezeichnet und die Leitungen für Port B mit dem Buchstaben "D". Dies dient zur Unterscheidung der Signale von IC PIO1. Für die nachfolgenden Versuche wird PAn mit PCn, PBn mit PDn, $\overline{\text{ASTB}}$ mit $\overline{\text{CSTB}}$, ARDY mit CRDY, BRDY mit DRDY und $\overline{\text{BSTB}}$ mit $\overline{\text{DSTB}}$ bezeichnet (n = 0 . . . 7). Die Anschlußstifte

mit Ausnahme der Port-Adreß-BUS-Leitungen und der Quittungsbetrieb-Leitungen sind auf der PC-Platine des Nanocomputers® fest verdrahtet. Aus Anhang D ist ersichtlich, welcher Pin mit welchem Anschluß an J7 verbunden ist. Der $\overline{CE}$-Pin von PIO2 ist mit der Leitung $\overline{IOQ2}$ verbunden, während die B/A- und C/D-Pins an die Adreßleitungen BA0 und BA1 angeschlossen sind. Führen die restlichen Adreßleitungen BA15 . . . BA2 die Logik-Signalfolge 0000 0000 0000 10, ist $\overline{IOQ2}$ aktiviert und PIO2 angewählt. Die Adreßleitung BA0 aktiviert entweder Port A oder Port B, während BA1 die übertragenen Daten als Steuer- oder Datenbytes definiert. Die Adressierungen für PIO1 und PIO2 sind in Tabelle 7-1 zusammengefaßt:

Tabelle 7-1. PIO-Portadressen des Nanocomputers®.

PIO	Port	Leitungen	Adressen (hex)
1	A, Daten	PA0 . . . PA7	04
	A, Steuerung	—	06
	B, Daten	PB0 . . . PB7	05
	B, Steuerung	—	07
2	A, Daten	PC0 . . . PC7	08
	A, Steuerung	—	0A
	B, Daten	PD0 . . . PD7	09
	B, Steuerung	—	0B

Zwecks Übereinstimmung mit den Unterlagen des Nanocomputers® ist Gatter A von PIO2 durch Gatter C und Gatter B durch Gatter D zu ersetzen.
Die Daten-BUS-Leitungen der Z-80-CPU sind direkt an dem PIO-Daten-BUS angeschlossen. Die PIO-Steuerleitungen $\overline{M1}$, $\overline{IORQ}$ und $\overline{RD}$ sind ebenfalls direkt mit den Anschlußstiften der Z-80-CPU verbunden. Der Z-80-Taktgeber ist am PIO-Takteingang (Pin 25) angeschlossen. Die Beschreibung der Interrupt-Steuerleitungen IEI, IEO und $\overline{INT}$ folgt zu einem späteren Zeitpunkt. An dieser Stelle nur soviel: sie werden in einer Verkettung von BUS-Leitungen mit PIO1 benutzt, die einen Prioritätsaufbau für die Handhabung von Interrupts von den Geräten bewirken.

Anmerkung: In den folgenden Versuchen sind Port A und Port B von PIO2 mit C und D bezeichnet. Das ist lediglich eine Vereinbarung, die sich nur auf den Nanocomputer® bezieht. Für die Z-80-PIO als Einzelbaustein gilt nach wie vor die Bezeicnung A und B.

Die nachfolgende Aufstellung gibt einen Überblick über den Inhalt der durchzuführenden Versuche:

Versuch Nr.	Bemerkungen
1	PIO-Betrieb nach Methode 0 ohne Signale für Quittungsbetrieb.
2	PIO-Betrieb nach Methode 0 mit Signalen für den Quittungsbetrieb.

3	Funktionsuntersuchung der Signale für Quittungsbetrieb bei PIO-Betrieb nach Methode 0.
4	PIO-Betrieb nach Methode 1.
5	PIO-Betrieb nach Methode 2.
6	PIO-Betrieb nach Methode 3.
7	Demonstration der relativen Prioritätsfolge für beide Ports innerhalb eines PIO-ICs.
8	Verkettungstechnik von PIO-Schaltungen.

VERSUCH NR. 1

Der erste Versuch demonstriert den Betrieb des PIO-ICs nach Methode 0 ohne Benutzung der Leitungen für den Quittungsbetrieb.

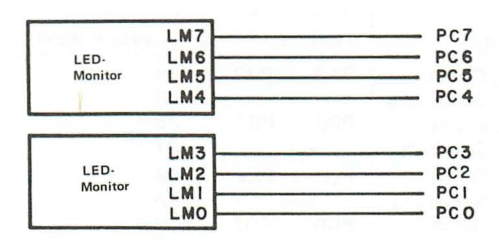


Bild 7-5. Anschluß der LED-Monitoren LM0 . . . LM7 an die Leitungen PC0 . . . PC7.

Programm OUTSIM

Objekt-Code		Quell-Code	Bemerkung
		NAME OUTSIM	
3E0F	OUTSIM:	LD A,0FH	; PIO2 auf Methode 0 programmie- ; ren
D30A		OUT (0AH),A	
3E43		LD A,43H	; Byte 43H an die Leitungen ; PC0 . . . PC7 ausgeben.
D308		OUT (08H),A	;
76		HALT	

1. Schritt

Die mit OUTSIM bezeichnete Routine laden. Der erste Ausgabebefehl gibt das Byte 0F an die Portadresse 0A. Die Portadresse 0A ist das Steuer-Port A von PIO2. (Laut Vereinbarung ist es für den Nanocomputer® Port C). Die vier im niedrigwertigen Halbbyte des Byte 0F gesetzten Bits D0 . . . D3 informieren den PIO, daß 0F ein Befehlsbyte zum Setzen der PIO-Betriebsart ist. Die beiden höchstwertigen Bits D6 und D7 bestimmen die Methode. Da die Bits D7 und D6 beide 0 sind, ist Methode 0 (Ausgabe) gewählt. Zur Erinnerung noch einmal das Steuerwort zum Setzen der Betriebsart:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

Die zweite Byteausgabe ist eine 43 an Port 08. Dabei handelt es sich um das Daten-Port A von PIO2 (oder Port C von PIO2). In Port C wird das Datenbyte wie bei einem Auffang-Register zwischengespeichert. Die Leuchtdioden zeigen zum Programmende die 43 an (es leuchten LM0, LM1 und LM6 auf).

2. Schritt

Lassen Sie das Programm ab Speicherstelle OUTSIM ablaufen.
Der LED-Monitor LM0 . . . LM7 zeigt 43 an. Das Programm hat nur einen Ausgabebefehl an Port C des PIO ausgeführt und dann angehalten (HALT). Da die Ausgabe mit Hilfe der LEDs weiter angezeigt wird, ist dies ein positiver Beweis für das Zwischenspeichern der Daten in Port C des PIO-ICs. Das mit dem PIO-IC über den Daten-BUS (Leitungen PC0 . . . PC7) verbundene externe Gerät braucht die Daten nicht "im Fluge" zwischenzuspeichern. Es hat genügend Zeit, die Daten zu lesen, wenn es dazu bereit ist.

3. Schritt

$\overline{\text{M1}}$ aktivieren durch kurzes Antippen von $\overline{\text{BM1}}$ an Masse mit Hilfe eines Drahtstückes. Ersetzen Sie anschließend den Befehl LD A,0FH durch zwei NOP-Befehle (bei OUTSIM und OUTSIM+1 00 eingeben). Dadurch entfallen die Programmierbefehle für PIO2. Lassen Sie das Programm ab OUTSIM ablaufen.
Der LED-Monitor leuchtet nicht auf und zeigt die Hexzahl 43 nicht an. Durch den Rücksetzimpuls hat PIO2 automatisch die Betriebsart nach Methode 1 gewählt. Dies geschieht immer, wenn durch logisch 0 an $\overline{\text{M1}}$ das IC zurückgesetzt wird. Das PIO nimmt ebenfalls nach dem Einschalten der Stromversorgung die Betriebsart nach Methode 1 ein. Diese Tatsachen machen deutlich, daß man PIO nach jedem Rücksetzimpuls neu programmieren muß, wenn das IC nicht nach Methode 1 arbeiten soll.
Der Versuch hat gezeigt, daß mit nur geringem externen Logikaufwand das PIO-IC die Funktion eines Ausgabe-Auffang-Registers ausübt. Dabei bleiben die Leitungen für den Quittungsbetrieb unbenutzt.

VERSUCH NR. 2

Der Versuch verwendet bei Operationen nach Methode 0 die Leitungen für den Quittungsbetrieb. Bauen Sie zusätzlich zu der Schaltung aus Bild 7-5 noch die Schaltung aus Bild 7-6 auf.

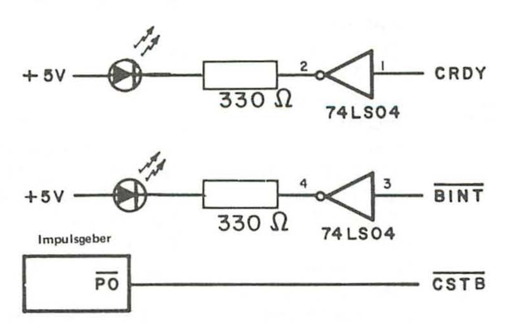


Bild 7-6. Anzeige für das CRDY- und für das $\overline{\text{BINT}}$-Signal.

Programme: INITOC, MAIN und SERVOC

Objekt-Code		Quell-Code	Bemerkung
		NAME INITOC	
ED5E	INITOC:	IM2	; Interrupt-Modus wählen.
21000F		LD HL, TABLE	; Adresse der Vektortabelle.
7C		LD A,H	; HI-Adreßbyte.
ED47		LD I,A	; Interrupt-Register setzen.
FD21E803		LD IY,SERVOC	; PIO-Ausgabe-Service-Routine.
FD22060F		LD (TABLE+06H),IY	; Vektortabelle einsetzen.
3E06		LD A,06H	; Interrupt-Vektor für
D30A		OUT (0AH),A	; für Port C laden.
08		EX AF,AF'	; Format für CONVDI laden.
3E40		LD A,40H	
08		EX AF,AF'	
3E0F		LD A,0FH	; PIO-Betriebsart wählen ; (Methode 0).
D30A		OUT (0AH),A	
3E87	ENPIO:	LD A,87H	; PIO-Interrupt freigeben.
D30A		OUT (0AH),A	
3EFF		LD A,0FFH	; CRDY-Signal auslösen.
D308	THROW:	OUT (08H),A	
C3C302		JP MAIN	; zur MAIN-Routine springen.

Objekt-Code		Quell-Code	Bemerkung
		NAME MAIN	
FB	MAIN:	EI	; Interrupt freigeben.
DD21000C		LD IX,DSTACK	; Boden des Daten-Stapelspeichers.
DD3600FF		LD (IX+00H),0FFH	; Zeitgeber für Anzeige.
21E50F		LD HL,ADDH	; Hinweisadresse zum Puffersetzen.
ED57		LD A,I	; Wert von IFF2 suchen.
EAD802		JP PE,HIGH	
3600	LOW:	LD (HL),00H	; Wert = 0.
1802		JR NEXT	
3610	HIGH:	LD (HL),10H	; Wert = 1.
2B	NEXT:	DEC HL	; Puffer-Zeiger schieben.
35		DEC (HL)	; COUNT erniedrigen.
ED73E20F		LD (DATAL),SP	; SP zum Puffer übertragen.
21B90F		LD HL,LEDL	; Für CONVDI einrichten.
11E50F		LD DE,ADDH	; Für CONVDI einrichten.
00	DISAB:	NOP	; Keine Operation.
CD7CFA		CALL CONVDI	
CD09F9	DLOOP:	CALL DISPL	
DD3500		DEC (IX+00H)	; Zeitgeber für Anzeige.
20F8		JR NZ,DLOOP	
C3C302		JP MAIN	; Rücksprung auf Routine MAIN.

Objekt-Code		Quell-Code	Bemerkung
		NAME SERVOC	
E5	SERVOC:	PUSH HL	; Inhalt der CPU-Register
F5		PUSH AF	; zwischenspeichern.
3AE40F		LD A,(ADDL)	; Pufferwert in A laden.
D308		OUT (08H),A	; Pufferwert ausgeben.
F1		POP AF	; Inhalt der CPU-Register
E1		POP HL	; umspeichern.
FB		EI	; Interrupt-Freigabe.
ED4D		RETI	; von Interrupt zurück.

1. Schritt

Schaltung aus Bild 7-6 aufbauen. Die Signale für den Quittungsbetrieb ARDY und $\overline{\text{ASTB}}$ sind zur Unterstützung von interruptgesteuerten I/O-Geräten bestimmt. Der Begriff interruptgesteuerte I/O bezieht sich auf die Art der Datenflußsteuerung zwischen der CPU und dem externen Gerät. Warum ist eine Datenflußsteuerung erforderlich? Betrachten Sie das folgende Programm, das 256 Datenbytes an ein externes Gerät ausgibt:

```
DUMP:   LD A,0FH        ; Betriebsart der PIO programmieren.
        OUT (0AH),A     ; Steuerung der Betriebsart an
                        ; Port A ausgeben.
        LD HL,0F800H    ; HL = Startadresse für Ausgabe-
                        ; Datenblock.
        LD B,0FFH       ; B = Bytezähler.
LOOP:   LD A,(HL)       ; ein Byte nach dem anderen
        OUT (08H),A     ; ausgeben.
        INC HL
        DJNZ LOOP
        HALT
```

Die Arbeitsfrequenz der CPU beträgt 2,5 MHz. Ist das PIO-IC programmiert, gibt das Programm DUMP in ca. 3,29 Millisekunden insgesamt 256 Bytes an ein angeschlossenes externes Gerät aus. Externe Geräte wie Kassettenrekorder, Drucker, Fernschreiber (TTY) und Kathodenstrahlröhren (CRT) arbeiten nicht einmal annährend so schnell. Nachstehend einige Beispiele:

TTY – Für den Einschreibvorgang der I/O-Operation benötigt ein Gerät 100 Millisekunden, das entspricht 10 Datenwörter in einer Sekunde (1 Datenwort = 10 Bit; die Übertragungsgeschwindigkeit ist also 100 Baud). Benötigt ein anderes Gerät 33,3 Millisekunden, ist die Übertragungsschwindigkeit 30 Wörter pro Sekunde oder 300 Baud.

CRT – Mögliche Übertragungsgeschwindigkeiten sind 300 Baud oder 24 500 Baud. Im letzteren Fall dauert der Einschreibvorgang 408,16 Mikrosekunden.

Für das Überschreiben der I/O-Operation besteht somit ein grundsätzliches Zeitproblem. Langsame externe Geräte müssen mit einer schnellen CPU zusammenarbeiten. Zur Lösung des Problems gibt es mehrere Möglichkeiten. Nachstehend zwei davon:

Methode 1: Zyklisches Abfragen des externen Gerätes durch die CPU
Bei dieser Methode fragt die CPU das externe Geräte ständig nach dessen Aufnahmebereitschaft ab. Die Antwort ist für die CPU wegen der relativ hohen Geschwindigkeitsdifferenz häufig negativ, bis sie ein positives Bereitschaftssignal empfängt. Ist das externe Gerät bereit, gibt die CPU das Byte aus. Bei dem Bereitsschaftssignal handelt es sich in der Regel um ein normales Bit oder ein externes Merkbit (Flag).

Methode 2: Interruptgetriebene I/O
Bei dieser Methode muß das externe Gerät von der CPU eine Unterbrechung anfordern, wenn es für das nächste Byte bereit ist. Nach Empfang der Interrupt-Anforderung gibt die CPU das nächste Datenbyte aus. Da die CPU im Unterschied zur Methode 1 das externe Gerät nicht zyklisch abfragt, führt sie nach Ausgabe des Datenbytes weitere Verarbeitsaufgaben durch (z.B. Errechnen des nächsten Ausgabe-Bytes).
Beim zyklischen Abfragen wird die Geschwindigkeitsdifferenz durch Verlangsamung der CPU ausgeglichen. Dadurch entsteht bei der CPU ein enormer Leistungsverlust. Ist z.B. das externe Gerät ein Standard-Fernschreiber mit einer Geschwindigkeit von 110 Baud, verringert das zyklische Abfragen die Arbeitsgeschwindigkeit der CPU um ca. 85%.

2. Schritt
In diesem Versuch verwenden Sie IC PIO2, um interruptgetriebene I/O mit der Z-80-CPU zu untersuchen. Die MAIN-Routine stellt Bearbeitungen dar, welche die Z-80 zwischen den Datenübertragungen durchführt. Ein Informationsaustausch zwischen dem Benutzer und der Z-80-CPU vollzieht sich nach folgendem Schema:

1. Die CPU ist mit der Bearbeitung beschäftigt.
2. Das externe Gerät fordert durch Erzeugung einer Unterbrechung eine Ausgabe von der CPU an.
3. Die CPU stellt die Bearbeitung vorübergehend ein und überträgt die Steuerung an eine Interrupt-Service-Routine.
4. Die Interrupt-Service-Routine gibt ein Datenbyte an das externe Gerät aus.
5. Die CPU nimmt die Bearbeitung wieder auf.

Bild 7-7 stellt die Ereignisfolge bei interruptgetriebenen I/O-Operationen dar.

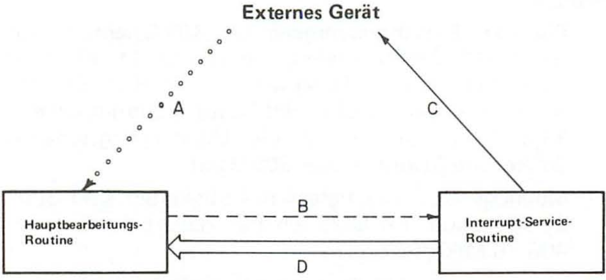


Bild 7-7. Interruptgetriebene Ausgabe. Pfad A stellt das Interrupt-Signal dar, Pfad B die Software-Übertragung der Steuerung als Antwort auf die Unterbrechung. Pfad C symbolisiert die Datenausgabe durch die Interrupt-Service-Routine und Pfad D die Rückkehr der Software-Steuerung zur Hauptbearbeitungs-Routine.

Schauen Sie sich den Vorgang in Bild 7-7 näher an, indem Sie die Sonderrolle des PIO-ICs und die Funktionen der Signale $\overline{ASTB}$, ($\overline{CSTB}$) und ARDY (CRDY) analysieren. Hierbei handelt es sich nur um eine von mehreren Möglichkeiten (siehe Anmerkung am Ende von Schritt 3).

1. Die CPU führt das Programm MAIN aus.
2. Das externe Gerät fordert eine Ausgabe vom PIO an, indem es die Leitung $\overline{CSTB}$ in den Zustand low versetzt.
3. Als Antwort auf die Erkennung des aktiven $\overline{CSTB}$-Signals aktiviert der PIO die Interrupt-Leitung der Z-80-CPU. Somit fordert der PIO ein Interrupt an. Gleichzeitig setzt das PIO-IC das CRDY-Signal zurück, damit das externe Gerät das Ausgabebyte nicht eher liest, bevor es bereit ist.
4. Die Z-80-CPU erkennt die Interrupt-Anforderung und sendet ein Interrupt-Bestätigungssignal zum PIO-IC zurück (aktiviert $\overline{M1}$ und $\overline{IORQ}$).
5. Das PIO-IC legt das LO-Byte der Vektor-Tabellenadresse für die Programmunterbrechung auf den Daten-BUS D0 . . . D7.
6. Die (für Unterbrechungen nach Methode 2 programmierte) CPU liest den Geräte-Erkennungscode ab und überträgt die Steuerung zur Interrupt-Service-Routine SERVOC.
7. Die Interrupt-Service-Routine SERVOC gibt ein Byte an Port A des PIO2 ab (Daten-BUS PC0 bis PC7).
8. Das IC PIO2 informiert das externe Gerät: das Ausgabebyte ist angekommen. Dazu nimmt Pin CRDY den Zustand logisch 1 ein.
9. Das externe Gerät kann das Byte jederzeit ablesen. Inzwischen hat die CPU die Ausführung des Programms MAIN wieder aufgenommen.

3. Schritt

Wie bereits erwähnt, besteht die von Ihnen zu verwendende Software aus drei Routinen: INITOC, MAIN und SERVOC. (Die MAIN-Routine ist bereits in Kapitel 6 beschrieben). Die Funktionen von INITOC und SERVOC sind folgende:

INITOC – Sie ist eine Initialisierungs-Routine für PIO2 zur Ausgabe an das Daten-Gatter (Port A); O steht für Ausgabe und C für Port A (beim (Nanocomputer® Port C), Leitungen PC0 . . . PC7.

INITOC erfüllt folgende Funktionen:

a) Setzen des Interrupt-Modus Methode 2.
b) Initialisieren des I-Registers zum HI-Byte der Interrupt-Vektortabellenadresse.
c) Laden der Interrupt-Service-Routine SERVOC in die Interrupt-Vektortabelle.
d) Interrupt-Vektor in das PIO-IC laden. Der Interrupt-Vektor ist der Geräte-Erkennungscode, den das PIO-IC bei einer Interrupt-Bestätigungsoperation der CPU auf den Daten-BUS legt. Dieses Byte wird unter Verwendung des folgenden Formats in das Steuergatter Port C von PIO2 geschrieben:

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

Weil das niederwertigste Bit "0" ist, übersetzt das PIO-IC das Steuer-

byte als Interrupt-Vektor. Bei diesem Versuch ist der Interrupt-Vektor 06.

e) Die PIO-Betriebsart auf Methode 0 setzen.

f) Das Interrupt-Steuerwort setzen, um Unterbrechungen freizugeben. Für den Betrieb nach Methode 0 hat das Interrupt-Steuerwort folgendes Format:

D7	D6	D5	D4	D3	D2	D1	D0
EI	X	X	X	0	1	1	1

wobei EI das Freigabe-Flag der Unterbrechung ist (EI = Enable Interrupt). Wird das EI-Bit gesetzt, gibt das Interrupt-Steuerwort Unterbrechungen für das PIO-Gatter frei. Ist also das Gattersignal $\overline{CSTB}$ durch ein externes Gerät aktiviert, fordert PIO mit ebenfalls aktivierten $\overline{INT}$-Signal bei der CPU ein Interrupt an. Beim $\overline{CSTB}$ ist die ansteigende (positive) Flanke für die Interrupt-Anforderung verantwortlich. Bild 7-8 zeigt die relative Zeitfolge des Signals für den PIO-Betrieb nach Methode 0.

Ist das EI-Bit zurückgesetzt, sind Unterbrechungen für die PIO-Ports blockiert. Das PIO-IC aktiviert das $\overline{INT}$-Signal also nicht, wenn das $\overline{CSTB}$-Signal von low auf high wechselt.

g) Ein unaufgefordertes Anfangsbyte ausgeben, damit das CRDY-Signal aktiv wird. Für einige Ausgabegeräte muß die CPU ein "Aufweck"-Byte aussenden, a) um das externe Gerät zu aktivieren und b), um das externe Gerät zu informieren, daß es Ausgabebytes anfordern kann, indem die $\overline{CSTB}$-Leitung vorübergehend in den Low-Zustand übergeht. Für "dumme" externe Geräte, die nicht zwischen einem "Aufweck"- und einem Datenbyte unterscheiden können, braucht die CPU kein Anfangsbyte auszugeben. Die Ausgabe des unaufgeforderten Bytes stellt aber eine gute Programmtechnik dar, um feste Betriebsregeln für das externe Gerät aufzustellen:

1) EIN AUSGABE-DATENBYTE SETZEN, ABER NUR WENN ARDY GESETZT IST.
2) DAS NÄCHSTE DATENBYTE ANFORDERN DURCH VORÜBERGEHENDES SETZEN VON $\overline{CSTB}$ IN DEN LOW-ZUSTAND.

Da das Anfangsbyte unaufgefordert erscheint, ignoriert das externe Gerät es. Alle folgenden Bytes werden vom externen Gerät mit eigener Geschwindigkeit angefordert und "ausgeführt", (z.B. gedruckt, gelocht, gespeichert).

SERVOC – Eine Interrupt-Service-Routine zur Ausgabe an Datengatter C (Port A) von PIO2. Wie bei INITOC haben O und C dieselbe Bedeutung. SERVOC hat folgende Funktionen:

a) den Zustand der CPU-Register H, L, A und F zwischenspeichern; da die SERVOC-Routine nur diese Register benutzt.

b) Den laufenden Wert des Zählers an das PIO-Datengatter Port C ausgeben. Der Zählerinhalt wird von der MAIN-Routine vermindert und an der Speicherstelle ADDL gespeichert.

c) Den CPU-Zustand umspeichern.

d) Interrupts freigeben. Denken Sie daran, maskierbare Interrupts werden blockiert, wenn die CPU eine Interrupt-Anforderung annimt und bestätigt.

e) Die Steuerung zur MAIN- (RETI) Routine zurückgeben. Der Steuerungsablauf zwischen den Routinen INITOC, MAIN und SERVOC ist schematisch in Bild 7-9 dargestellt.

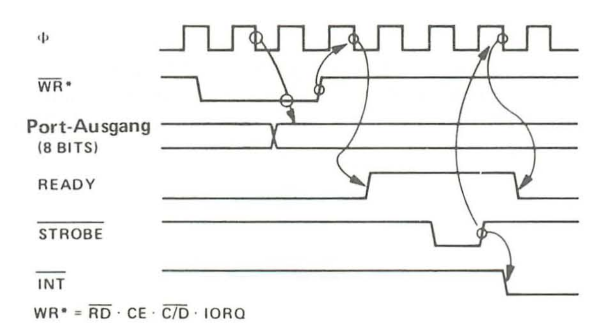


Bild 7-8. Zeitdiagramm für den Z-80-PIO-Versuch nach Methode 1.

Anmerkung: Vom momentanen Thema abweichend ist an dieser Stelle das PIO RDY/STB-Quittungsbetriebsprotokoll beschrieben. Wenn externes Gerät STB in den Low-Zustand versetzt, ist dies eine Anforderung für das nächste Datenbyte. In diesem Zusammenhang wird das erste von der CPU ausgegebene unaufgeforderte Byte als ein AUFWECK-Byte übersetzt. Nachstehend eine alternative Übersetzung des PIO-Quittungsbetriebsprotokolls, bei der das erste unaufgeforderte Byte als Datenbyte interpretiert werden kann.
Versetzt ein externes Gerät STB in den Low-Zustand, ist dies eine BESTÄTIGUNG für den Empfang des letzten Bytes.
Bei dieser Interpretation startet die CPU die Aktion mit dem ersten DATEN-Byte. Die Geschwindigkeit für die weitere Kommunikation hängt von der Leistungsfähigkeit des externen Gerätes ab, alle nachfolgenden Byteausgaben von der CPU zu bestätigen.
Die Übersetzung des BESTÄTIGUNGS- und ANFORDERUNGS-Protokolls ist identisch, sieht man von der Bedeutung des ersten unaufgeforderten Bytes ab.

4. Schritt

Starten Sie das Programm ab Speicherstelle INITOC. Das Programm führt die MAIN-Routine aus. Sie erkennen das an der bereits bekannten Anzeige (Zähler mit abnehmenden Inhalt). Dabei ist das Signal CRDY aktiviert (logisch 1). Optisch deutlich wird dieser Zustand durch Aufleuchten der mit CRDY verbundenen LED. Die LEDs LM0 . . . LM7 zeigen das erste unaufgeforderte Ausgabebyte an, das die CPU an das Datengatter Port C des PIO-ICs ausgegeben hat.

5. Schritt

In diesem Schritt übernehmen Sie selbst die Aufgabe eines an Datengatter Port C angeschlossenen externen Gerätes. Wenn Sie für die Entgegennahme eines Datenbytes bereit sind, müssen Sie es bei der CPU anfordern. Dazu

betätigen Sie den Impulsgeber $\overline{PO}$; das Anforderungssignal $\overline{CSTB}$ ist aktiviert.
Infolge der Anforderung wird CRDY logisch 1 und die LEDs LM0 . . . LM1 zeigen den momentanen Zählerinhalt an.

6. Schritt
$\overline{PO}$ in Arbeitsstellung halten. Die Anzeige des LED-Monitors (LM0 . . . LM7) bleibt unverändert. $\overline{PO}$ in die Ruhestellung zurückkehren lassen.
Fast gleichzeitig mit der Freigabe von $\overline{PO}$ zeigt der LED-Monitor den aktuellen Zählerinhalt an.

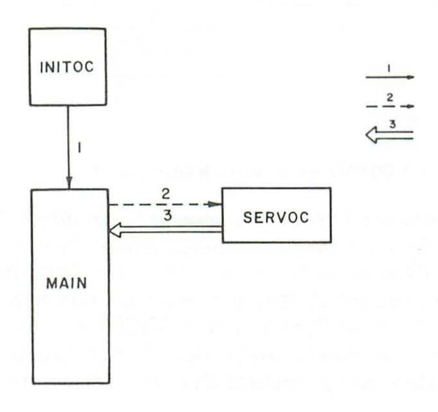


Bild 7-9. Steuerungsablauf zwischen den Routinen INITOC, MAIN und SERVOC.
1 = Beendigung der Routine INITOC
2 = Antwort auf Unterbrechung
3 = Rückkehr von Unterbrechung

7. Schritt
Obwohl in Schritt 6 ein Interrupt erfolgt ist (Anzeige des LED-Monitors), ändert sich am Anzeigezustand von CRDY und $\overline{BINT}$ nichts. Um den Grund herauszufinden, prüfen Sie einmal die Zeitfolge für den PIO-Betrieb nach Methode 0.

Mit der Aktivierung der Abtastimpulsleitung $\overline{STROBE}$ ($\overline{ASTB}$ oder $\overline{CSTB}$) treten folgende Ereignisse ein:

1. Mit der ansteigenden Flanke des $\overline{STROBE}$-Signals aktiviert der PIO seine $\overline{INT}$-Leitung die über $\overline{BINT}$ gepuffert und mit der CPU verbunden ist. Bei der CPU ist somit eine Unterbrechung angefordert.
2. Mit der nächsten abfallenden Flanke des Taktimpulses deaktiviert der PIO die READY- (ARDY oder CRDY) Leitung und zeigt somit an, die Ausgabedaten sind noch nicht bereit.
3. Die CPU bestätigt die Unterbrechung durch Aktivierung von $\overline{INTA}$. Dementsprechend wird das für das PIO-IC interne Interrupt-Flipflop zurückgesetzt, woraufhin das $\overline{INT}$-Signal in den High-Zustand (logisch 1) zurückkehrt.

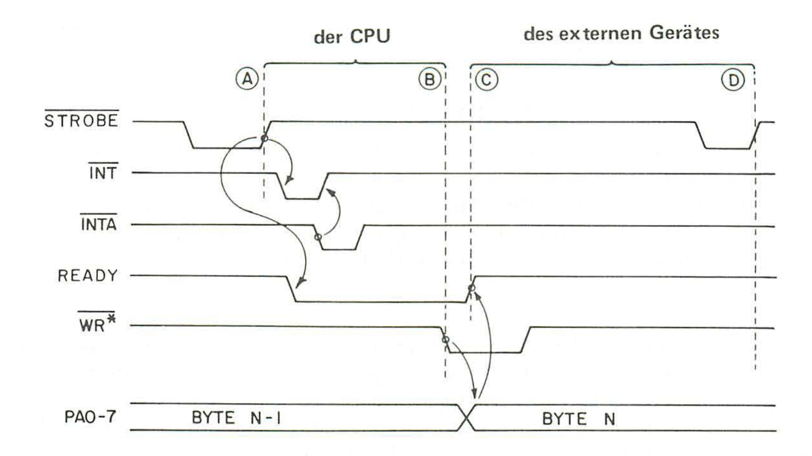


Bild 7-10. Zeitdiagramm für den Z-80-PIO-Betrieb nach Methode 0. An Punkt A fordert das externe Gerät ein Byte an. An Punkt B beginnt die CPU einen Ausgabezyklus zur Ausgabe des nächsten Bytes. Bei Punkt C teilt der PIO dem externen Gerät die Bereitschaft des Ausgabebytes mit. Bei Punkt D fordert das externe Gerät ein weiteres Byte an, wodurch der Abrufzyklus erneut beginnt.

4. Wenn die CPU die Steuerung zur Interrupt-Service-Routine SERVOC gibt, wird ein Ausgabebefehl für das PIO-Datengatter ausgeführt. Die während des Ausgabezyklus aktivierten CPU-Signale schließen sich zusammen, um das $\overline{\text{WR}}$*-Signal zu aktivieren (Erklärung folgt).
5. Mit der Aktivierung des $\overline{\text{WR}}$*-Signals liest der PIO das Ausgabe-Datenbyte vom CPU-Daten-BUS D0 . . . D7 in das PIO-Ausgabe-Datenregister.
6. Mit der nächsten abfallenden Flanke des Taktimpulses aktiviert der PIO das READY-Signal und zeigt damit an, daß die Ausgabedaten zum Ablesen durch das externe Gerät bereit sind.

Die $\overline{\text{INT}}$-Leitung ist daher nur solange logisch 0, bis die CPU sie abtastet und bestätigt, indem sie dem PIO ein $\overline{\text{INTA}}$-Signal sendet ($\overline{\text{M1}}$ und $\overline{\text{IORQ}}$ sind aktiviert). Von der Interrupt-Anforderung bis zum OUT-Befehl im Interrupt-Service ist die CRDY-Leitung ein paar Befehlszyklen lang im Low-Zustand. Diese Änderung haben Sie deshalb nicht bemerkt, weil das menschliche Auge und auch die LEDs auf derartige schnelle Zustandsänderungen zu träge reagieren.

ERKLÄRUNG DES $\overline{\text{WR}}$*-SIGNALS

Das $\overline{\text{WR}}$*-Signal ist ein low-aktives Signal, das intern vom PIO-IC erzeugt wird. Wie bereits bekannt, hat der PIO nur Anschlußstifte für $\overline{\text{IORQ}}$, $\overline{\text{M1}}$ und $\overline{\text{RD}}$. Der $\overline{\text{WR}}$-Ausgang an der Z-80-CPU ist nicht mit dem PIO-IC verbunden. Wenn also die CPU dem PIO ein Datenbyte ausschreibt, zieht das IC aus den logischen Zuständen von $\overline{\text{RD}}$, $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ und C/D entsprechende Schlußfolgerungen. $\overline{\text{WR}}$* ist nur dann aktiv, wenn ($\overline{\text{RD}}$ NICHT

aktiv ist) und ($\overline{CE}$ aktiv ist) und (C/D im Zustand logisch 1 ist, der ein Datenbyte anzeigt) und ($\overline{IORQ}$ aktiv ist).
Leider wird dieses Verhältnis manchmal wie folgt geschrieben:

$$\overline{WR}^* = \overline{RD} \cdot \overline{CE} \cdot C/D \cdot \overline{IORQ}.$$

Diese Schreibweise ist nicht korrekt, da sie positive und negative Logik miteinander verbindet.
Konzentrieren Sie sich deshalb nur darauf, welche Signale aktiv sein müssen, damit auch $\overline{WR}^*$ aktiv ist. Die logischen Zustände für $\overline{RD}$, $\overline{CE}$, C/D und $\overline{IORQ}$ sind in der genannten Reihenfolge 1010.

8. Schritt
Fordern Sie noch einige Bytes von der CPU an. Dabei stellen Sie fest, daß die CPU nur angeforderte Bytes ausgibt. Zwischen den Datenübertragungen setzt die CPU die unabhängige Bearbeitung der MAIN-Routine fort. Sie erkennen die an der sich laufend ändernden 7-Segmentanzeige. Fordern Sie mehrere Bytes möglichst schnell hintereinander bei der CPU an. Erkennen Sie einen Leistungsabfall der CPU, indem z.B. der angezeigte Zählerinhalt sich langsamer ändert?
Nein, die CPU *scheint* zwei Funktionen gleichzeitig auszuführen: an das externe Gerät die geforderten Bytes auszugeben und den Zählerinhalt zu dekrementieren. Natürlich fällt die Leistung leicht ab, weil für die Durchführung der Interrupt-Service-Routine SERVOC Zeit verstreicht. Die Abnahme ist aber sehr geringfügig und daher nicht wahrnehmbar.

9. Schritt
Nanocomputer® zurücksetzen (RESET-Taste drücken). INITOC-Routine wie folgt ändern:

Speicherstelle	ändern von	ändern zu
THROW	D3	00
THROW + 1	08	00

Dadurch entfällt der Befehl, der das Aufweckbyte ausgibt und das CRDY-Signal aktiviert. Starten Sie das Programm erneut bei INITOC. Die LEDs LM0 . . . LM7 bleiben dunkel und registrieren ein Byte, das 00 ist. Da noch kein Byte an das PIO-IC ausgegeben ist, bleibt das Ausgabe-Datenregister Port C vom PIO inaktiv. Bei dem angezeigten Byte 00 braucht es sich also nicht um das geänderte Aufweckbyte zu handeln. Auch die LED CRDY bleibt dunkel. Falls das externe Gerät – in diesem Fall also Sie – der strengen Regel folgt
NUR EIN DATENBYTE LESEN, WENN CRDY AKTIV IST,
warten Sie vergebens. Ist also eine solche Regel überhaupt sinnvoll? Die Antwort ist ja. Diese Regel hindert das externe Gerät daran, dasselbe Ausgabebyte zweimal oder ein falsches Byte zu lesen. Tritt zwischen der Byte-Anforderung des externen Gerätes ($\overline{CSTB}$ logisch 0) und der Ankunft beim PIO eine längere Verzögerung ein, ist das CRDY-Signal ein sehr wichtiger Indikator für das externe Gerät. Beispiel: Fordert das externe Gerät ein Byte an, versetzt es $\overline{CSTB}$ in den Low-Zustand. Liest nun das externe Gerät das Byte am PIO-Datengatter *bevor* CRDY *aktiv ist,* handelt es sich um das bereits bei der ersten Anforderung vorhandene Byte. Bei

einer weiteren Anforderung würde das bereits gelesene Byte noch einmal gelesen.
Da CRDY nicht aktiv ist, das externe Gerät aber ein Byte anfordern muß, "um die Dinge in Gang zu bringen", bildet dies eine Ausnahme zu der strengen Regel. Ein "genügend intelligentes" externes Gerät kann in die Lage versetzt werden, die Ausnahme zu erkennen und die Regel danach streng zu befolgen. Für weniger "intelligente externe" Geräte verwendet man besser ein Aufweckbyte, um CRDY in den High-Zustand zu versetzen; so braucht sich das externe Gerät nicht um Ausnahmen zu kümmern.
Da Sie im laufenden Versuch die Funktion eines externen Gerätes ausüben, können Sie also die Ausnahme erkennen. Fordern Sie ein Datenbyte an, indem Sie das Signal $\overline{CSTB}$ in den Low-Zustand versetzen. Von da an ist das Anfordern und Empfangen von Daten wieder völlig problemlos.

VERSUCH NR. 3

Der Versuch demonstriert die Operation des CRDY-Signals im Quittungsbetrieb. Dazu ist eine Verzögerung zwischen der Interrupt-Erzeugung durch das PIO-IC und der Ausgabe eines Datenbytes an das PIO-IC durch die Interrupt-Service-Routine erforderlich. Zu diesem Zwecke wird die SERVOC-Routine gegen die SEROCX-Routine ausgetauscht. Die für diesen Versuch notwendigen Schaltungen entnehmen Sie den Bildern 7-5 und 7-6.

Programme: INITOC, MAIN und SEROCX
Die Programme INITOC und MAIN sind bereit bekannt, deshalb folgt an dieser Stelle nur das Programm-Listing von SEROCX.

Objekt-Code		Quell-Code	Bemerkung
		NAME SEROCX	
C5	SEROCX:	PUSH BC	; Inhalt der CPU-Register
D5		PUSH DE	; zwischenspeichern
E5		PUSH HL	
F5		PUSH AF	
DDE5		PUSH IX	
FDE5		PUSH IY	
FD2AE40F		LD IY,(ADDL)	; Zustand von ADDL zwischen-
FDE5		PUSH IY	; speichern.
DD23	DSX:	INC IX	; Stackpointer aktualisieren.
DD23		INC IX	
DD23		INC IX	
00		NOP	; keine Operation.
DD3600FF		LD (IX+00H),0FFH	; DLOOPX-Zeit setzen.
DD36010A		LD (IX+01H),00AH	; CLOOPX-Zeit setzen.
DD360201	CLOOPX:	LD (IX+02H),01H	; DLOOPX-Zeit setzen.
21E50F		LD HL,ADDH	; auf Anzeigepuffer hinweisen.
ED57		LD A,I	; Wert von IFF2 feststellen.
EA0E06		JP PE,HIGHX	
3600	LOWX:	LD (HL),00H	; Wert = 0

```
1802                JR NEXTX
3610     HIGHX:     LD (HL),10H        ; Wert = 1
2B       NEXTX:     DEC HL             ; Pufferzeiger verschieben.
34                  INC (HL)           ; ADDL erhöhen.
ED73E20F            LD (DATAL),SP      ; Stackpointerinhalt zum
21B90F              LD HL,LEDL         ; Puffer übertragen.
11E50F              LD DE,ADDH         ; für CONVDI setzen.
CD7CFA              CALL CONVDI        ; für CONVDI setzen.
CD09F9   DLOOPX:    CALL DISPL
DD3500              DEC (IX+00)        ; Zeitgeber für Anzeige.
20F8                JR NZ,DLOOPX
DD3502              DEC (IX+02)        ; Zeitgeber für Anzeige
20F3                JR NZ,DLOOPX
DD3501              DEC (IX+01)        ; Zeitgeber für Service-Routine.
20CD                JR NZ,CLOOPX
FDE1                POP IY             ; CPU-Registerinhalt zurückspeichern.
FD22E40F            LD (ADDL),IY       ; Zustand von (ADDL) umspeichern.
3AE40F   OUTX:      LD A,(ADDL)        ; Byte ausgeben, das beim
D308                OUT (08H),A        ; Interrupt im ADDL war.

FDE1                POP IY             ; CPU-Registerinhalte zurück-
DDE1                POP IX             , speichern.
F1                  POP AF
E1                  POP HL
D1                  POP DE
C1                  POP BC
FB                  EI                 ; Interrupt freigeben.
ED4D                RETI               ; von Interrupt zurückkehren.
```

1. Schritt

Dieser Versuch ist eine Fortsetzung des Versuchs Nr. 2. Falls Sie Versuch Nr. 2 noch nicht durchgeführt haben, holen Sie es an dieser Stelle nach.

Die SEROCX-Routine ist der SERVOC-Routine sehr ähnlich, sie führt folgende Funktionen aus:

1. Zwischenspeichern der CPU-Registerinhalte.
2. Ablaufvariantes Programm gewährleisten durch Zurverfügungstellung eines neuen Datenraums für die Zählerschleifen.
3. Verzögerung von ca. 10 Sekunden einfügen, während der Zählerinhalt erhöht wird.
4. Ausgabe des augenblicklichen Zählerinhaltes an den LED-Monitor.
5. Umspeicherung des CPU-Zustands.
6. Rückgabe der Steuerung an die MAIN-Routine.

Um SERVOC gegen SEROCX als Interrupt-Service-Routine auszutauschen, INITOC-Routine wie folgt ändern:

Speicherstelle	ändern von	ändern in
INITOC + 10	niederwertiges Byte (LO-Byte) der SERVOC-Adresse	LO-Byte der SEROCX-Adresse
INITOC + 11	hochwertiges Byte (HI-Byte) der SERVOC-Adresse	HI-Byte der SEROCX-Adresse

Nicht vergessen, das Aufweckbyte wieder in die INITOC-Routine einzusetzen. Prüfen Sie besonders, ob die folgenden Speicherstellen die nachstehenden Werte haben:

Speicherstelle	Wert
THROW	D3
THROW + 1	08

2. Schritt
Ausführung bei INITOC beginnen. Impulsgeber $\overline{P0}$ schalten. Der vom Tastenfeld -Display angezeigte Zählerinhalt hört auf sich zu vermindern und beginnt sich zu erhöhen. Die Leuchtdiode CRDY erlischt. Nach 10-facher Erhöhung des Zählers treten die folgenden drei Ereignisse gleichzeitig ein:

- CRDY geht auf logisch 1 und die LED leuchtet auf.
- Beim Interrupt-Signal wird der Inhalt des Zählers auf dem LED-Monitor LM0 . . . LM7 angezeigt.
- Die MAIN-Routine übernimmt wieder die Steuerung der CPU. Auf dem Display erscheint die Stackpointer-Adresse 0F00 und der Zählerinhalt wird wieder dekrementiert (erniedrigt).

Wichtig ist die enorm lange Verzögerungszeit zwischen der neuen Byteanforderung (Schalten des $\overline{P0}$) und der Ankunft des neuen Bytes. Während dieses Intervalls wird das CRDY-Signal deaktiviert, damit das externe Gerät das PIO-Datengatter nicht abliest, sondern wartet. Um richtig mit dem PIO-IC zusammenzuwirken, muß in dem externen Gerät diese Wartelogik vorhanden sein.

VERSUCH NR. 4

Versuch Nr. 4 zeigt den PIO-Betrieb nach Methode 1. Ein nach Methode 1 arbeitendes PIO-Gatter gleicht einem Eingabe-Port mit Puffern und Auffangregistern. Das PIO-IC bietet aber noch ein zusätzliches Merkmal, nämlich die interruptbetriebene Eingabe, die unter Verwendung der zwei Quittungsbetriebssignale $\overline{STB}$ und RDY realisiert wird.

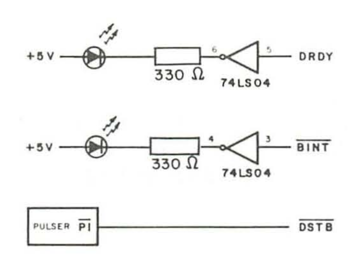


Bild 7-11. Schaltung für den PIO-Versuch nach Methode 1.

Programme: MAIN, INITID, SERVID und SERVI
Neben der MAIN-Routine sind für diesen Versuch noch folgende Programme erforderlich:

Objekt-Code		Quell-Code	Bemerkung
		NAME INITID	
ED5E	INITID:	IM2	; Interrupt-Modus 2.
21000F		LD HL,TABLE	; Adresse der Vektortabelle.
7C		LD A,H	; HI-Adreß-Byte.
ED47		LD I,A	; Interruptregister setzen.
FD211F04		LD IY,SERVID	; Eingabe-Service-Routine.
FD22080F		LD (TABLE+08H),IY	; Vektortabelle einsetzen.
3E08		LD A,08H	; Interrupt-Vektor laden.
D30B		OUT (0BH),A	
08		EX AF,AF'	; Format für CONVDI setzen.
3E40		LD A,40H	
08		EX AF,AF'	
3E4F		LD A,4FH	; PIO-Methode wählen.
D30B		OUT (0BH),A	
3E87		LD A,87H	; PIO-Interrupt freigeben.
D30B		OUT (0BH),A	
DB09		IN A,(09H)	; DRDY setzen.
C3C302		JP MAIN	

Objekt-Code		Quell-Code	Bemerkung
		NAME SERVID	
C5	SERVID:	PUSH BC	
0E09		LD C,09H	; Port D Interrupt
C33104		JP SERVI	

ANMERKUNG: Die Routine SERVI entnehmen Sie dem Anhang B.

1. Schritt
Schaltung aus Bild 7-11 anfertigen. Es sind zwei LED-Schaltungen vorgesehen, um den Zustand von $\overline{\text{DSTB}}$ und DRDY anzuzeigen. In diesem Versuch benötigen Sie die Daten-Gatterleitungen PD0 . . . PD7 von PIO2, Port B.

2. Schritt
INITID und SERVID sind die neuen Routinen für diesen Versuch. (I bedeutet Input (Eingabe) und D bezeichnet bei PIO2 Port D des Nanocomputers®). Bild 7-12 zeigt den Steuerungsablauf zwischen den Routinen INITID, MAIN, SERVID und SERVI. Bild 7-13 ist das Zeitdiagramm.

INITID: Diese Initialisierungsroutine führt folgende Funktionen aus:

1. Setzen der CPU-Interrupt-Bearbeitung auf Methode 2.
2. Initialisierung des I-Registers.
3. Laden der Interrupt-Service-Routine SERVID in die Interrupt-Vektortabelle.
4. Ausgabe des Interrupt-Vektors an das Steuergatter von PIO2, Port B, d.h. PIO-Gatter Port D.
5. Setzen von Port D für den Betrieb nach der Eingabemethode

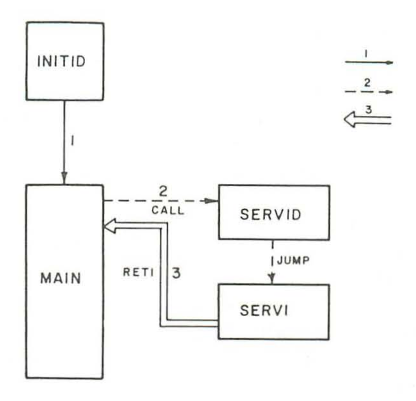


Bild 7-12. Steuerungsablauf für den PIO-Versuch nach der Eingabemethode.
1 = Beendigung der INITID-Routine
2 = Antwort auf Unterbrechung
3 = Rückkehr von Unterbrechung

Nr. 1; das geschieht durch Ausgabe des folgenden Bytes zur Gatteradresse 0B, Port D:

0100	1111
4	F

Die vier niederwertigen Bits definieren das gesamte Byte als Methoden-Auswahlbyte, während die beiden höchstwertigen Bits die Betriebsart (Methode 0 . . . 3) bestimmen.

6. Interrupt-Freigabe für Port D; dies geschieht durch Ausgabe des folgenden Bytes an Adresse 0B:

1000	0111
8	7

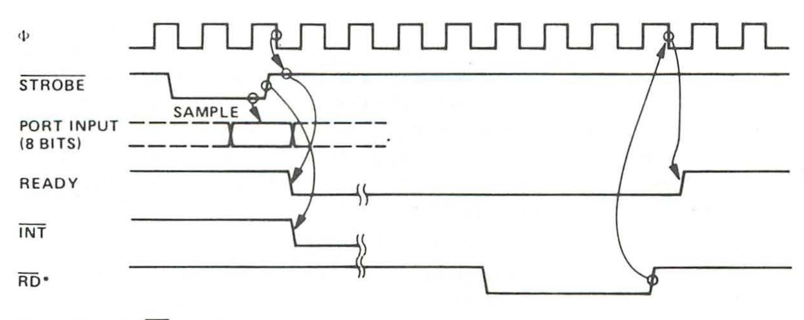


Bild 7-13. Zeitdiagramm für PIO-Eingabemethode Nr. 1.

Die vier niederwertigen Bits zeigen, daß es sich um ein Interrupt-Steuerungsbyte handelt. Beim Betrieb nach Methode 1 wird als einziges Bit D7, das Interrupt-Freigabebit, benutzt. Durch das Setzen von Bit D7 werden Unterbrechungen freigegeben. Das PIO-Interrupt-Freigabeflipflop arbeitet vollständig unabhängig von IFF1 und IFF2 der Z-80-CPU.

7. DRDY auf logisch 1 setzen (aktivieren), indem ein Aufweckbyte an PIO2-Port B auf den Leitungen PD0 . . . PD7, das als Gatter 09 adressiert ist, eingegeben wird.

SERVID: Eine Interrupt-Service-Routine mit folgenden Funktionen:

1. Zwischenspeichern der CPU-Registerinhalte.
2. Zwischenspeichern des augenblicklichen Zählerwertes, der an der Speicherstelle ADDL gespeichert ist.
3. Eingabe des Datenbytes vom Datengatter D des PIO-Port B.
4. Laden des Eingabebytes in die Speicherstelle ADDL zur anschließenden Anzeige.
5. Schaffung eines neuen Datenraums für die Zeitbytes, die dieses Programm zu einem ablaufvarianten Programm machen.
6. Anzeige des Eingabebytes anstelle des Zählerinhaltes für etwa etwa 10 Sekunden.
7. Umspeichern des CPU- und Zählerinhaltes in Speicherstelle ADDL.
8. Interrupts freigeben.
9. Rückgabe der Steuerung an das MAIN-Programm.

3. Schritt

Mit der Aktivierung des Signals $\overline{\text{STROBE}}$ ($\overline{\text{BSTB}}$ oder $\overline{\text{DSTB}}$) ereignet sich folgendes: (siehe Bild 7-13)

1. Das PIO-IC liest die Daten vom externen Gerät in ein internes Daten-Eingaberegister.
2. Die ansteigende Flanke des Abtastimpulses läßt den PIO das $\overline{\text{INT}}$-Signal aktivieren, das von der CPU eine Unterbrechung anfordert.
3. Die nächste abfallende Flanke des Taktes versetzt das READY-Signal in den Low-Zustand. Das bedeutet, die CPU hat das externe Eingabebyte noch nicht gelesen. So ist das READY-Signal auch für Eingabeoperationen kritisch. Es zeigt dem externen Gerät an, das PIO-Eingaberegister ist voll und soll nicht eher geladen werden, bis die CPU es gelesen hat.
4. Die CPU überträgt die Steuerung an die Interrupt-Service-Routine.
5. Die Interrupt-Service-Routine führt einen IN-Befehl des PIO-Datengatters aus, der folgende Signale aktiviert:

   $\overline{\text{RD}}$, $\overline{\text{CE}}$ und $\overline{\text{IORQ}}$

   und außerdem nimmt die C/D-Leitung den High-Zustand (logisch 1) ein. Dadurch wird das $\overline{\text{RD}}^*$-Signal aktiviert. Am Ende der Leseoperation ist die READY-Leitung aktiviert.
6. Sobald die READY-Leitung aktiv ist, kann das externe Gerät jederzeit ein weiteres Byte von der CPU abfordern.

4. Schritt

Stellen Sie mit den Schaltern SW0 . . . SW7 die Hex-Ziffer 07 ein. Starten Sie anschließend das Programm ab Speicherstelle INITID. Das 7-Segment-Display zeigt die bekannte Anzeige für die MAIN-Routine; außerdem leuchtet LED $\overline{\text{BINT}}$ auf. Weil die INITID-Routine ein Aufweckbyte erzeugt, leuchtet ebenfalls die DRDY-LED. Impulsgeber P1 von der Ruhe- in die Arbeitsstellung schalten und dort festhalten. Auf der gesamten Anzeige (7-Segment-Display und LEDs) dürfen keine Veränderungen eintreten. Impulsgeber P1 in die Ruhestellung zurückkehren lassen.
Auf dem 7-Segment-Display erscheint der Inhalt der Logikschalter SW0 . . . SW7 07 anstelle des Zählerinhalts für ca. 10 Sekunden. Während dieser Zeit ist IFF2 zurückgesetzt (die erste Anzeige von links zeigt 0), weil maskierbare Interrupts blockiert sind. Die Adresse im Stapelzeiger (Stack) wechselt von 0F00 nach 0EEC. Das entspricht insgesamt 12 Bytes, die der Stapelspeicher aufnimmt.

ANMERKUNG: Möglicherweise (wenn auch sehr unwahrscheinlich) kann der Stapelspeicher anstatt nach 0EEC auch nur nach 0EF0 geschoben werden. Warum?

5. Schritt

Die Byte-Einstellung der Logikschalter ändern und einen weiteren $\overline{\text{DSTB}}$-Impuls erzeugen. Das eingestellte Schalterbyte erscheint für ca. 10 Sekunden auf der Anzeige. Wiederholen Sie diesen Vorgang mit mehreren verschiedenen Eingabebytes, so daß Ihnen die Operationen der Signale $\overline{\text{DSTB}}$ und DRDY im Quittungsbetrieb deutlich werden.

6. Schritt

Stellen Sie mit SW0 . . . SW7 wieder 07 ein und erzeugen einen $\overline{\text{DSTB}}$-Impuls. Lösen Sie während des Interrupt-Vorganges einen weiteren $\overline{\text{DSTB}}$-Impuls aus.

ANMERKUNG: Die Interrupt-Service-Routine SERVID wird nicht in abhängigen Speicherstellen gespeichert. SERVID besteht aus zwei Codesegmenten. Das erste Segment ist aus 6 Bytes zusammengesetzt, deren Eintrittspunkt mit SERVID bezeichnet wird. SERVID benutzt den Stack (Stapelspeicher) für die Zwischenspeicherung des BC-Registerinhaltes und lädt dann die Gatteradresse 0CH) des Datengatters D von PIO-Port B in das C-Register. Anschließend übernimmt die SERVI-Routine die Steuerung. Das bei der Speicherstelle SERVI beginnende abhängige Codesegment ist das zweite oben erwähnte Codesegment. SERVI ist ABLAUFVARIANT. Die Gatteradresse gelangt durch das C-Register zu diesem Codesegment. Somit kann man das Codesegment SERVI für jede Gatteradresse benutzen, die man zu lesen wünscht. Ändert man den NOP-Befehl an der Speicherstelle ENABG (in der SERVI-Routine) in den EI-Befehl zur Freigabe des Z-80 Interrupt-Flipflops (Hex-Code FB), ist das Codesegment für die Bearbeitung jeder Gatteradresse geeignet. Insbesondere entsteht so für jedes Eingabegatter eine Interrupt-Service-Routine, die vom gegebenen Gatter liest. Mit diesen Routinen lassen sich viele Interrupts wirksam umsetzen. Jede dieser Programmunterbrechungen beginnt mit 6 Bytes, ähnlich wie die 6 Bytes der SERVID-Routine. Die Steuerung übernimmt dann das Codesegment SERVI. Es ist ablaufvariant und kann viele externe Geräte bedienen, die eine Bearbeitung anfordern. Prüfen Sie nun die Routinen SERVID, SERVIC, SERVIE und SERVIF.

Der zweite $\overline{\text{DSTB}}$-Impuls setzt das DRDY-Signal zurück. Infolgedessen erscheint die 07 auf der Anzeige für ca. 20 Sekunden. Die Anzeige dauert also doppelt solange gegenüber einer einzigen Unterbrechung.

7. Schritt
Stellen Sie nochmals 07 ein und übertragen diese Zahl mit Hilfe des Impulsgebers P1 auf die Anzeige. Während der Interrupt-Bearbeitung die Logikschalter auf 00 einstellen und P1 erneut auslösen.
Das DRDY-Signal und die $\overline{\text{BINT}}$-Leitung gehen direkt nach dem zweiten $\overline{\text{DSTB}}$-Signal auf logisch 0. Die ersten 10 Sekunden lang wird 07 angezeigt. Dann nimmt die CPU die zweite Unterbrechung an und bestätigt sie, wodurch das $\overline{\text{BINT}}$-Signal auf high (logisch 1) geht. Anschließend zeigt die Anzeige 10 Sekunden lang 00. Die beiden Unterbrechungen werden also SERIELL bearbeitet, eine direkt nach der anderen. Warum findet keine Verschachtelung beider Unterbrechungen statt?

Die SERVID-Routine blockiert bis zum vorletzten Befehl die Interrupts. Erst dann sind die Interrupt-Signale frei. Das PIO-IC speichert in einem interen Interrupt-Flipflop das Signal, bevor es nach Beendigung der ersten Interrupt-Bearbeitung erkannt, angenommen und bedient wird.

8. Schritt
Wie viele Unterbrechungen speichert das PIO-IC bis zur Wiederannahme von maskierbaren Interrupts durch die CPU? Um dies herauszufinden, erzeugen Sie mehrere $\overline{\text{DSTB}}$-Signale in schneller Folge. Wie viele Unterbrechungen bedient die CPU?
Sie können aus der Anzeige erkennen, daß die CPU nur zwei Interrupts bearbeitet. Das 7-Segment-Display zeigt 20 Sekunden das 00-Byte. Folglich hat das PIO-IC eine ähnliche innere Schaltung, wie das von Ihnen aufgebaute Flipflop in Versuch Nr. 1 aus Kapitel 6. Beide können nur eine Zustandsänderung speichern.

VERSUCH NR. 5

Der fünfte Versuch zeigt den PIO-Betrieb nach Methode 2 (bidirektionale Methode). Diese Methode arbeitet mit Port A zusammen und benutzt sowohl die Quittungsbetriebssignale A als auch B. Für die Datenausgabe wird Signal A, für die Dateneingabe Signal B benutzt. Damit Port A nach Methode 2 arbeiten kann, muß Port B für den Betrieb nach Methode 3 programmiert sein. Wie bereits mehrfach erwähnt, ist Port A von PIO2 auf der Nanocomputer®-Experimentierplatine Port C und Port B ist Port D.

Programme: MAIN, SERVOC, SERVIC und INITPB
Neben den bereits bekannten Routinen MAIN und SERVOC werden in diesem Versuch noch SERVIC und INITPB benutzt.

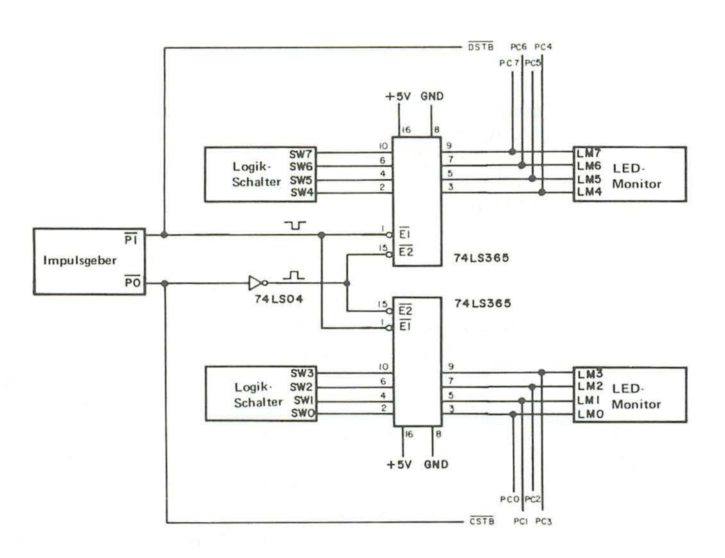


Bild 7-14. Schaltung für den PIO-Betrieb nach Methode 2.

Objekt-Code		Quell-Code	Bemerkung
		NAME SERVIC	
C5	SERVIC:	PUSH BC	; BC zwischenspeichern
0E08		LD C,08H	; Interrupt Port C
C33104		JP SERVI	

Objekt-Code		Quell-Code	Bemerkung
		NAME INITPB	
ED5E	INITPB:	IM2	; Interrupt-Modus 2
21000F		LD HL,TABLE	; Adresse der Vektortabelle
7C		LD A,H	; HI-Adreßbyte
ED47		LD I,A	; Interrupt-Register setzen
FD21E803		LD IY,SERVOC	; Ausgabe Service-Routine
FD22060F		LD (TABLE+06H),IY	; Vektortabelle einsetzen
FD211904		LD IY,SERVIC	; Eingabe Service-Routine
FD220A0F		LD (TABLE+0AH),IY	; Vektortabelle einsetzen
3E06		LD A,06H	; Interrupt-Vektor für
D30A		OUT (0AH),A	; Port C laden
3E0A		LD A,0AH	; Interrupt-Vektor für
D30B		OUT (0BH),A	; Port D laden
08		EX AF,AF'	; Format für CONVDI setzen
3E40		LD A,40H	
08		EX AF,AF'	
3E8F		LD A,8FH	; PIO-Methode 2 für
D30A		OUT (0AH),A	; Port C setzen
3ECF		LD A,0CFH	; PIO-Methode 3 für
D30B		OUT (0BH),A	; Port D setzen
3EFF		LD A,0FFH	; Maskenbyte für Port D setzen
D30B		OUT (0BH),A	; um zu folgen, PIO-Methode 3
			; setzen

```
3E87      LD A,87H       ; PIO-Interrupt freigeben
D30A      OUT (0AH),A    ; Gatter C
D30B      OUT (0BH),A    ; Gatter D
3EFF      LD A,0FFH      ; CDRY initialisieren (setzen)
D308      OUT (08H),A
DB08      IN A,(08H)     ; DRDY initialisieren
C3C302    JP MAIN        ; zur MAIN-Routine springen
```

1. Schritt

In diesem Versuch programmieren Sie PIO2 (PIO-Port C), um nach der bidirektionalen Methode 2 zu arbeiten. Methode 2 ist im wesentlichen eine Kombination der auf einem Port überlagerten Eingabe- und Ausgabemethoden. Port A wird benutzt, um Daten in beiden Richtungen zwischen dem PIO-IC und der CPU zu übertragen. Die Quittungsbetriebssignale $\overline{\text{ASTB}}$ und ARDY synchronisieren Byte-Ausgabeoperationen, während $\overline{\text{BSTB}}$ und BRDY Eingabeoperationen koordinieren. Die grundsätzlichen Quittungsbetriebsregeln sind denen für Methode 0 und Methode 1 sehr ähnlich mit einer wichtigen Ausnahme. Bild 7-15 zeigt die Zeitfolge für Operationen nach Methode 2. Sobald das $\overline{\text{ASTB}}$-Signal aktiviert ist, treten folgende Ereignisse ein:

1. Während $\overline{\text{ASTB}}$ (in diesem Versuch $\overline{\text{CSTB}}$) aktiv ist, hält der Daten-BUS von Port A (in diesem Versuch die Leitungen PC0 bis PC7) den aktuellen Inhalt des Ausgaberegisters von Port A fest. Das externe Gerät muß dieses Byte lesen, während $\overline{\text{ASTB}}$ aktiv ist. Sobald $\overline{\text{ASTB}}$ nicht mehr aktiviert ist, steht das Signal nicht mehr auf dem Daten-BUS von Port A zur Verfügung.
2. Mit der ansteigenden Flanke von $\overline{\text{ASTB}}$ erzeugt der PIO eine Interrupt-Anforderung an die Z-80-CPU.
3. Das ARDY-Signal (in diesem Versuch CRDY) wird deaktiviert. Es zeigt dadurch an: das nächste Ausgabebyte ist noch nicht bereit, vom externen Gerät abgelesen zu werden.
4. Die Interrupt-Leitung $\overline{\text{INT}}$ der CPU ist im Low-Zustand. Während des Interrupt-Anforderungs-/Bestätigungszyklus legt das PIO-IC das LO-Byte der Vektortabellenadresse für die Ausgabe-Interrupt-Service-Routine auf den Daten-BUS der CPU. Das PIO-IC benutzt den Ausgabe-Interrupt-Vektor, weil $\overline{\text{ASTB}}$ den Zustand logisch 0 angenommen hat, womit ein weiteres Ausgabebyte angefordert wird.
5. Die Ausgabe-Interrupt-Service-Routine führt einen OUT-Befehl an Port A aus und aktiviert das $\overline{\text{WR}}$*-Signal.
6. Mit der nächsten abfallenden Flanke des Taktes nach der ansteigenden Flanke des $\overline{\text{WR}}$*-Signals wird die ARDY-Leitung aktiv. Dadurch ist das nächste Ausgabebyte bereit.

Die vorstehende Ereignisfolge beschreibt den Ausgabeteil der bidirektionalen Methode. Bevor Sie sich mit dem Eingabeteil befassen, sei auf einen kritischen Unterschied zwischen der vorigen und der Methode 0 hingewiesen. *Die Ausgabedaten sind nur solange auf dem Daten-BUS von Port A, wie $\overline{\text{ASTB}}$ aktiv ist.* Beim Betrieb nach Methode 0 wird das Ausgabebyte unabhängig vom $\overline{\text{ASTB}}$-Zustand auf dem Daten-BUS gehalten und erlaubt somit dem externen Gerät das Byte nach Belieben zu lesen. Da man die

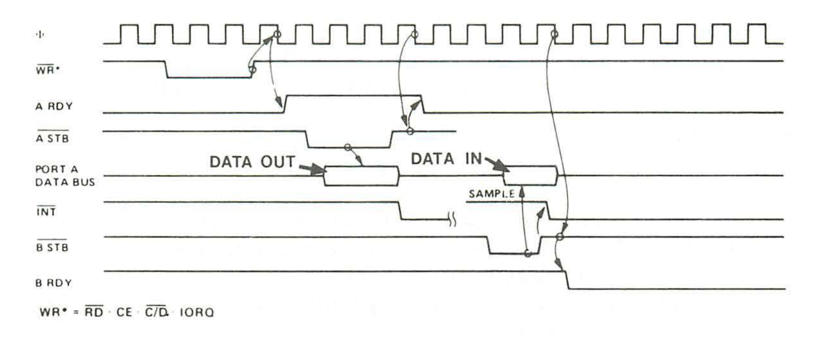


Bild 7-15. Zeitdiagramm für die PIO-Eingabemethode Nr. 2 (bidirektionale Methode).

Datenleitungen von Port A sowohl für Ein- als auch Ausgaben benutzen kann, dürfen die Ausgabedaten nicht wie beim Betrieb nach Methode 0 ständig die Leitungen blockieren. Folglich werden die Daten für die Leitungen nur freigegeben, wenn das externe Gerät ein weiteres Ausgabebyte anfordert; wenn $\overline{ASTB}$ aktiviert ist. Das hat wichtige Konsequenzen im Hinblick auf die ein Ausgabebyte anfordernde Logik:

a) Um ein Ausgabebyte zu lesen, muß das externe Gerät das nächste Byte anfordern.
b) Das vom externen Gerät gelesene Byte hat die CPU beim Interrupt-Service als letztes Byte ausgegeben.
c) Das externe Gerät bleibt stets ein Byte "hinter" der CPU zurück.

Untersuchen Sie nun die Ereignisfolge bei einer Byte-Eingabeoperation. Mit Aktivierung des $\overline{BSTB}$-Signals treten folgende Ereignisse ein:

1. Die Eingabedaten werden mit der abfallenden Flanke von $\overline{BSTB}$ für den Daten-Bus von Port A freigegeben (in diesem Versuch $\overline{DSTB}$).
2. Während $\overline{BSTB}$ aktiv ist (low), liest das PIO-IC die Eingabedaten und erzeugt eine Interrupt-Anforderung an die CPU.
3. Mit der nächsten abfallenden Flanke des Taktes deaktiviert der PIO das BRDY-Signal (in diesem Versuch DRDY), um dem externen Gerät anzuzeigen: das PIO-Eingabegatter ist voll, kein weiteres Byte einlesen.
4. Die CPU tastet eine aktivierte $\overline{INT}$-Leitung und führt einen Interrupt-Anforderungs-/Bestätigungszyklus aus.
5. Nach dem Empfang einer Interrupt-Bestätigung ($\overline{INTA}$ aktiv, wenn $\overline{M1}$ und $\overline{IORQ}$ aktiv), legt der PIO das LO-Byte der Interrupt-Vektortabellenadresse für die Eingabe-Interrupt-Service-Routine auf den Daten-BUS der CPU.
6. Die CPU leitet die Steuerung an die Eingabe-Interrupt-Service-Routine weiter, die einen IN-Befehl ausführt, um das Datenbyte von dem Eingabe-Datenregister des PIO-Port A zu lesen.
7. Während des I/O-Lesezyklus wird das $\overline{RD}^*$-Signal in den Zustand low versetzt (aktiv). Die ansteigende Flanke von $\overline{RD}^*$ aktiviert die BRDY-Leitung. Dies signalisiert dem externen Gerät, das die CPU das Eingabebyte gelesen hat und zur Aufnahme eines weiteren Bytes bereit ist.

2. Schritt

In diesem Versuch werden die Programme INITPB, MAIN, SERVOC und SERVID benutzt. Die Programme MAIN und SERVOC sind bereits bekannt.

INITPB: Diese Initialisierungsroutine führt folgende Funktionen aus:

a) Setzen der Interrupt-Methode auf Modus 2.
b) Initialisierung von Register I.
c) Laden der SERVOC-Adresse in die Interrupt-Vektortabelle.
d) Laden der SERVIC-Adresse in die Interrupt-Vektortabelle.
e) Laden des Interrupt-Vektors vom PIO-Port C. Da die Quittungsbetriebssignale von Port C zur Byteausgabe gehören, weist der Interrupt-Vektor auf die SERVOC-Routine hin.
f) Laden des Interrupt-Vektors vom PIO-Port D. Da die Quittungsbetriebssignale von Port D zur Byteeingabe gehören, weist der Interrupt-Vektor auf die SERVIC-Routine hin.
g) Programmierung des PIO-Port C auf Betrieb nach Methode 2.
h) Programmierung des PIO-Port D auf Betrieb nach Methode 3. Da die Quittungsbetriebsleitungen des Ports D von Port C benutzt werden, ist die Betriebsmethode 3 für Port D die einzige Alternative.
i) Bestimmung des Maskenbytes für Port D. Das ist für den Betrieb nach Methode 3 eine Notwendigkeit, auch wenn es nicht benutzt wird.
j) Freigabe von Interrupts für Port C und D.
k) Initialisierung von CRDY und DRDY.

SERVIC: Die Interrupt-Service-Routine gehört zum PIO-Interrupt als Antwort auf die Aktivierung des $\overline{\text{DSTB}}$-Signals. SERVIC ist mit der SERVID-Routine identisch, nur erfolgt die Eingabe über Port C anstatt über Port D. Nachstehend die wichtigsten Funktionen:

a) Eingabe der Logik-Schalter-Information von Port C.
b) Anzeige der Eingabebytes für eine Zeit von 10 Sekunden auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers®.
c) Rückgabe der Programmsteuerung zur MAIN-Routine, die den Zählerinhalt weiter dekrementiert.

Bild 7-16 zeigt, wie SERVIC und SERVOC bei der Ein- und Ausgabefunktion zusammenarbeiten.

Durch Aktivierung des $\overline{\text{DSTB}}$-Signals erfolgen die Datenübertragungen I und II.

ÜBERTRAGUNG I: Die Daten der Logik-Schalter werden in das Eingaberegister von Port C eingegeben und eine Unterbrechung erzeugt.

ÜBERTRAGUNG II: Die Interrupt-Service-Routine SERVIC gibt die Daten der Logik-Schalter vom Eingaberegister des PIO-Port C ein und speichert sie in der Speicherstelle ADDL zur anschließenden Anzeige.
Durch Aktivierung des $\overline{\text{CSTB}}$-Signals erfolgen die Datenübertragungen III und IV.

ÜBERTRAGUNG III: Der Inhalt des PIO-Ausgaberegisters wird auf den Daten-BUS von Port C gelegt, damit das externe Gerät die Information lesen kann.

ÜBERTRAGUNG IV: Mit der ansteigenden Flanke des $\overline{\text{CSTB}}$-Signals

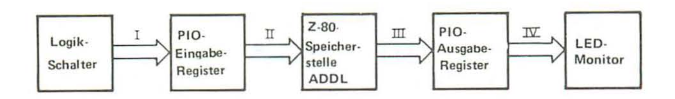


Bild 7-16. Datenübertragung bei der SERVIC- und SERVOC-Routine.

erzeugt das PIO-IC eine Interrupt-Anforderung an die CPU. Die CPU bestätigt die Unterbrechung und überträgt die Steuerung zur SERVOC-Routine. Die SERVOC-Routine gibt den Inhalt der Speicherstelle ADDL zum Ausgaberegister von Port C. Dieses Byte liest das externe Gerät nicht eher, bis ein aktiviertes $\overline{\text{CSTB}}$-Signal das neue Byte abruft.

3. Schritt

Bauen Sie die in Bild 7-14 dargestellte Schaltung auf. Der Impulsgeber $\overline{\text{P0}}$ fordert Ausgabebytes vom PIO-IC an, während Impulsgeber $\overline{\text{P1}}$ Eingabebytes für das PIO-IC anfordert. Die Impulsgeber sind an die Puffer-ICs 74LS365 angeschlossen. So gelangen die Daten der Logik-Schalter nur dann auf den Daten-BUS von Port A, wenn $\overline{\text{P1}}$ aktiv ist (logisch 0) und $\overline{\text{P0}}$ inaktiv (logisch 1). Das Schalten von $\overline{\text{P1}}$ allein erzeugt einen $\overline{\text{DSTB}}$-Impuls, gibt aber die Puffer nicht frei. Gleichzeitiges Schalten beider Impulsgeber ist weder ratsam noch gibt es die Puffer frei.

4. Schritt

Beginnen Sie die Programmausfphrung bei INITPB. Es passiert folgendes
a) CRDY und DRDY sind beide aktiv (high).
b) Die LEDs LM0 . . . LM7 sind dunkel.

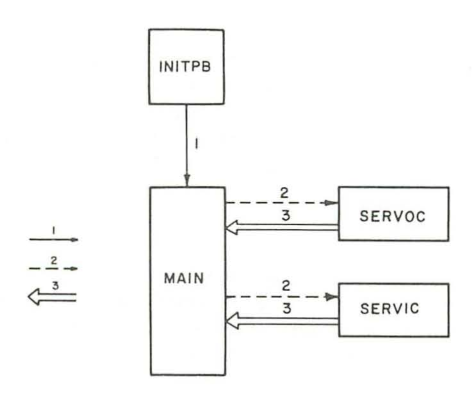


Bild 7-17. Steuerungsablauf zwischen den Routinen INITPB, MAIN, SERVOC und SERVIC.
1 = Beendigung der Routine INITPB
2 = Antwort auf Interrupt
3 = Rückkehr von Interrupt

c) Das Programm führt die MAIN-Routine aus. Der Zählerinhalt wird dabei pro Sekunde zweimal dekrementiert.
Bringen Sie die $\overline{\text{CSTB}}$-Leitung auf logisch 0, indem Sie den Impulsgeber $\overline{\text{P0}}$ aus seiner normalen Ruhestellung fern halten.
Betätigen Sie den Impulsgeber $\overline{\text{P0}}$ und halten ihn in Arbeitsstellung, so daß die $\overline{\text{CSTB}}$-Leitung auf logisch 0 geht und diesen Zustand beibehält.
Der LED-Monitor LM0 . . . LM7 zeigt das Aufweckbyte FF. Lassen Sie den Impulsgeber in die Ruhestellung zurückkehren und notieren dabei den Zählerinhalt auf der Nanocomputer®-Anzeige.
Sobald der Impulsgeber seine Ruhestellung einnimmt, geht $\overline{\text{CSTB}}$ auf logisch 1. Die ansteigende Flanke dieses Signals löst einen Interrupt-Impuls aus. Die SERVOC-Routine übernimmt infolgedessen die Steuerung der CPU. Der aktuelle Zählerinhalt gelangt zum Ausgabe-Register von Port C. Welchen Wert zeigt der LED-Monitor an? Nichts! Der Wert ist 00, so daß der LED-Monitor dunkel bleibt. Der Grund hierfür ist, daß sich die Daten-BUS-Leitungen PC0 . . . PC7 von Port C im hochohmigen Zustand befinden. Erst ein erneutes $\overline{\text{CSTB}}$-Signal gibt die Daten von Port C frei. Das ist der Hauptunterschied zwischen der Byteausgabe nach der bidirektionalen Methode und dem Betrieb nach Methode 0.

5. Schritt
Bringen Sie $\overline{\text{CSTB}}$ auf low und halten Sie diesen Zustand mit Hilfe des Impulsgebers P0 bei. Was geschieht jetzt?
Der bei der Erzeugung des ersten $\overline{\text{CSTB}}$-Impulses festgehaltene Wert kommt auf dem LED-Monitor LM0 . . . LM7 zur Anzeige.

6. Schritt
Wiederholen Sie die Schritte 4 und 5 mehrmals. Nur so erkennen Sie das Verhältnis zwischen dem *angezeigten* Byte und dem Zählerinhalt bei Eintritt der *vorigen* Unterbrechung.

7. Schritt
03 an den externen Schaltern einstellen. $\overline{\text{DSTB}}$ mit Hilfe des Impulsgebers P1 auf low bringen. Was sehen Sie?
03 wird sofort auf dem LED-Monitor angezeigt. Die Anzeige des Nanocomputers® bleibt unverändert, es wird der dekrementiernde Zählerinhalt angezeigt. Das bedeutet: die MAIN-Routine steuert die CPU immer noch.

8. Schritt
Lassen Sie den Impulsgeber P1 in seine Normalstellung zurückkehren. Dadurch entsteht beim $\overline{\text{DSTB}}$-Signal eine ansteigende Flanke. Das ausgelöste Interrupt-Signal gibt die Steuerung der CPU zur SERVIC-Routine zurück. Für ca. Sekunden erscheint auf der Anzeige anstelle des Zählerinhaltes das Byte 03.

9. Schritt
$\overline{\text{P0}}$ einmal auslösen und dabei die CRDY-LED beobachten.
Erkennen Sie ein Flackern?
Nein, obwohl CRDY bekanntlich zwischen der Erzeugung der Unter-

brechung und der Ausführung des OUT-Befehls durch die SERVOC-Routine auf low geht.
$\overline{P1}$ auslösen und sorgfältig die DRDY-LED beobachten. Auch hier ist keine Veränderung erkennbar, obwohl DRDY ebenfalls zwischen der Erzeugung der Unterbrechung und der Ausführung des IN-Befehls durch die SERVIC-Routine auf low geht.

10. Schritt
Stellen Sie mit den Logik-Schaltern 05 ein und betätigen einmal $\overline{P1}$. Dadurch entsteht ein $\overline{DSTB}$-Signal. Während 05 auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers® angezeigt wird, ein zweites $\overline{DSTB}$-Signal auslösen.
DRDY geht auf low. Das PIO-IC speichert die noch ausstehende Unterbrechung. Das erste Interrupt-Signal zeigt das Byte 05 ca. 10 Sekunden an. Daran anschließend folgt ohne Verzögerung die Bearbeitung der zweiten Unterbrechung. DRDY bleibt weiterhin auf logisch 0 und das Byte 05 für weitere 10 Sekunden auf der Anzeige. Die Bearbeitung der Unterbrechung erfolgt also seriell und nicht geschachtelt. Dies geht nicht nur aus dem Verhalten des DRDY-Signals hervor, sondern auch aus den angezeigten Werten des Stapelzeigers (Stack).

11. Schritt
$\overline{CSTB}$ mit Impulsgeber $\overline{P0}$ mehrmals auslösen. Zeigt CRDY jemals eine Veränderung?
Nein, Sie nehmen keine Änderung der CRDY-LED wahr; sie scheint ständig im Zustand logisch 1 zu bleiben. Die Bearbeitung der Interrupt-Service-Routine ist wesentlich schneller, als Sie den Impulsgeber auslösen können. Die SERVOC-Routine hat keine 10 Sekunden-Schleife wie die SERVIC-Routine.

12. Schritt
Logik-Schalter auf 06 setzen. Impulsgeber $\overline{P1}$ einmal auslösen, um das Byte einzugeben. Merken Sie sich den aktuellen Zählerinhalt, während Sie den Impulsgeber auslösen. Mit 06 auf der Anzeige Impulsgeber P0 auslösen, um ein Ausgabebyte anzufordern. Was beobachten Sie?
CRDY geht auf logisch 0, was eine Verzögerung zwischen der Anforderung und der Lieferung eines Ausgabebytes anzeigt. Für die Verzögerung ist die SERVIC-Routine verantwortlich. Sie benötigt einmal eine lange Ausführungszeit und blockiert zum anderen maskierbare Interrupts. Ist die SERVIC-Routine einmal ausgeführt, folgt die Bearbeitung der ausstehenden Unterbrechung, die CRDY wieder aktiviert. Um das durch Auslösen von P0 angeforderte Ausgabebyte zu beobachten, müssen Sie $\overline{CSTB}$ auf low halten. Das Ausgabebyte muß mit dem notierten Zählerinhalt übereinstimmen.

VERSUCH NR. 6

Die PIO-Betriebsart nach Methode 3 zeigt der Versuch Nr. 6. Die sogenannte Steuermethode unterscheidet sich von den anderen Betriebsarten, weil sie keine Quittungsbetriebssignale benutzt. Sie befaßt sich außerdem mit der Ein- und Ausgabe von Bits anstatt von Bytes.

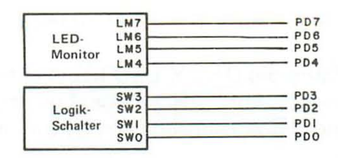


Bild 7-18. Verdrahtungsschema für den PIO-Versuch nach Methode 3.

Programme: MAIN, INITPM und SERVM

Neben der MAIN-Routine werden in diesem Versuch die folgenden Routinen benutzt:

Objekt-Code		Quell-Code	Bemerkung
		NAME INITPM	
ED5E	INITPM:	IM2	; Interrupt Modus 2
21000F		LD HL,TABLE	; Adresse der Vektortabelle
7C		LD A,H	; HI-Adreß-Byte
ED47		LD I,A	; Interrupt-Register setzen
FD210505		LD IY,SERVM	; Adresse der Service-Routine
FD220C0F		LD (TABLE+0CH),IY	; Verktortabelle einsetzen
3E0C		LD A,0CH	; Interrupt-Vektor für
D30B		OUT (0BH),A	; Port D setzen
08		EX AF,AF'	; Format für CONVDI setzen
3E40		LD A,40H	
08		EX AF,AF'	
3ECF		LD A,0CFH	; Methode 3 für Port D setzen
D30B		OUT (0BH),A	
3E0F		LD A,0FH	; Eingabeleitungen für
D30B		OUT (0BH),A	; Port D bestimmen
3E97	CWORD:	LD A,97H	; Interrupt-Steuerwort setzen
D30B		OUT (0BH),A	
3EFC		LD A,0FCH	; PB0, PB1 überwachen
D30B		OUT (0BH),A	
0E09		LD C,(09H)	; LED-Monitor in Position AUS,
3E00		LD A,00H	; d.h. alles LEDs sind dunkel
ED79		OUT (C),A	
C3C302		JP MAIN	

Objekt-Code		Quell-Code	Bemerkung
		NAME SERVM	
C5	SERVM:	PUSH BC	; Inhalt der CPU-Register
D5		PUSH DE	; zwischenspeichern
E5		PUSH HL	
F5		PUSH AF	
DDE5		PUSH IX	
FDE5		PUSH IY	
FD2AE40F		LD IY,(ADDL)	; Zustand von (ADDL) zwischen-
			; speichern
FDE5		PUSH IY	

```
0E09                LD C,09H            ; Eingabe von PIO-Port C
ED78                IN A,(C)
E60F                AND 0FH             ; die vier höherwertigen Bits
                                        ; auf Null setzen
32E40F              LD (ADDL),A         ; Byte in ADDL geben
17                  RLA                 ; Höherwertiges Halbbyte (die
17                  RLA                 ; vier höchstwertigen Bits)
17                  RLA                 ; durch niederwertiges Halbbyte
17                  RLA                 ; ersetzen
ED79                OUT (C),A
DD23       DSM:     INC IX              ; Stack-Pointer aktualisieren
DD23                INC IX
DD23                INC IX
00                  NOP                 ; keine Operation
DD3600FF            LD(IX+00H),0FFH     ; interne DLOOPM-Zeit setzen
DD36010A            LD (IX+01H),00AH    ; CLOOPM-Zeit setzen
DD360202 CLOOPM:    LD (IX+02H),02H     ; externe DLOOP-Zeit setzen
21E50F              LD HL,ADDH          ; auf Anzeigepuffer hinweisen
ED57                LD A,I              ; Wert von IFF2 feststellen
EA4105              JP PE,HIGHM
3600       LOWM:    LD (HL),00H         ; Wert = 0
1802                JR NEXTM
3610       HIGHM:   LD (HL),10H         ; Wert = 1
ED73E20F NEXTM:     LD (DATAL),SP       ; SP-Inhalt zum Puffer übertragen
21B90F              LD HL,LEDL          ; für CONVDI setzen
11E50F              LD DE,ADDH          ; für CONVDI setzen
CD7CFA              CALL CONVDI
CD09F9     DLOOPM:  CALL DISPL
DD3500              DEC (IX+00)         ; Zeitgeber für Anzeige
20F8                JR NZ,DLOOPM
DD3502              DEC (IX+02)         ; Zeitgeber für Anzeige
20F3                JR NZ,DLOOPM
DD3501              DEC (IX+01)         ; Zeitgeber für Service-Routine
20CF                JR NZ,CLOOPM
FDE1                POP IY              ; Inhalt von ADDL umspeichern
FD22E40F            LD (ADDL),IY
FDE1                POP IY              ; Inhalt der CPU-Register
DDE1                POP IX              ; umspeichern
F1                  POP AF
E1                  POP HL
D1                  POP DE
C1                  POP BC
FB                  EI                  ; Interrupts freigeben
ED4D                RETI                ; von Interrupt zurückkehren
```

1. Schritt

Bei der Eigenschaftsbestimmung des PIO-Port räumt der Betrieb nach Methode 3 einen großen Flexibilitätsspielraum ein. Daher müssen mit Hilfe von Befehlbytes verschiedene Alternativen ausgewählt und für das PIO-IC bestimmt werden. Prüfen Sie anhand der INITPM-Routine, wie ein PIO-Port für den Betrieb nach Methode 3 programmiert ist.

INITPM: Initiierungs-Routine, die neben anderen Funktionen PIO-Port D für den Betrieb nach Methode 3 programmiert. Alle INITPM-

Funktionen sind in der nachstehenden Aufstellung erwähnt. Richten Sie ein besonderes Augenmerk auf Funktionen für Methode 3:

a) Setzen der CPU-Interrupt-Methode auf Modus 2.
b) Initialisieren des I-Registers.
c) Laden der SERVM-Adresse in die Interrupt-Vektortabelle.
d) Interrupt-Vektor an PIO-Port D (PIO2, Port B) ausgeben.
e) PIO-Port D für Betrieb nach Methode 3 programmieren.
   Dazu ist folgendes Steuerbyte an das Steuergatter von Port D erforderlich:

1100	1111
C	F

   Der Gerätecode ist 0B. Das niederwertige Halbbyte ist ein Methoden-Auswahlbyte, während die Bits D7 und D6 die Methode 3 wählen.
f) Bestimmung des I/O-Auswahlregisters. Falls das Methoden-Auswahlbyte die Methode 3 festlegt, wird von dem nächsten Steuerbyte die Festlegung des I/O-Auswahlregisters erwartet. Das I/O-Auswahlregister besteht aus 8 Bits mit folgender Bedeutung:
   Falls das Bit Dn logisch 1 ist, bestimmt dieser Pegel die n-te Leitung PDn des PIO-Gatter-Daten-BUS zur EINGABE-Leitung. Ist Dn logisch 0, wird die n-te Leitung PDn des PIO-Gatter-Daten-BUS zur AUSGABE-Leitung bestimmt.
   INITPM gibt z.B. folgendes Byte aus:

   0000 1111

   Danach sind PD7, PD6, PD5 und PD4 als Ausgabe- und PD3, PD2, PD1 und PD0 als Eingabe-Leitungen bestimmt.
g) Setzen des Interrupt-Steuerworts: Für den Betrieb nach Methode 3 müssen folgende Bits bestimmt werden:
   a) *Das Interrupt-Freigabebit:* Unterbrechungen werden freigegeben, wenn das Bit gesetzt ist. INITPM setzt dieses Bit.
   b) Das AND/OR-Bit: Ist dieses Bit gesetzt, müssen alle unmaskierbaren Leitungen des Gatter-Daten-BUS aktiv sein, damit der PIO eine Unterbrechung erzeugt.
      Ist dieses Bit zurückgesetzt, braucht nur eine der unmaskierbaren Leitungen aktiv zu sein, damit der PIO eine Unterbrechung erzeugt. INITPM setzt dieses Bit zurück.
   c) Das HIGH/LOW-Bit: Dieses Bit legt den aktiven Zustand für die Gatter-BUS-Leitungen fest. Führt das HIGH/LOW-Bit den Wert logisch 1, ist high der aktive Zustand aller BUS-Leitungen. Bei logisch 0 sind die BUS-Leitungen low-aktiv.
      INITPM setzt dieses Bit zurück.
   d) Das Bit "Maske folgt": Ist dieses Bit gesetzt, (logisch 1), überträgt der PIO das nächste Befehlsbyte zum Gatter als Maskenbyte. INITPM setzt dieses Bit.
   e) Die Bits D3, D0 sind auf 0111 gesetzt und zeigen an: das Steuerbyte bestimmt das Interrupt-Steuerwort.

Zusammengefaßt hat das Interrupt-Steuerwort folgendes Format:

Interrupt Freigabe	AND/ OR	High/ Low	Masken- byte folgt	0	1	1	1

Die INITPM-Routine bestimmt das Interrupt-Steuerbyte, damit die unmaskierbaren Daten-BUS-Leitungen des PIO-Port D auf das Vorhandensein einer logischen 0 kontrolliert werden. Wird eine logisch 0 erkannt, hat dies ein Interrupt-Signal zur Folge.

h) Bestimmung des Maskenbytes: Das nächste von INITPM an Port D ausgegebene Steuerbyte ist das Maskenbyte (D4-Bit im Interrupt-Steuerwort). Dieses Byte bestimmt, welche Gatter-Daten-BUS-Leitungen maskierbar oder nichtmaskierbar sind. Jedes Bit im maskierten Byte bei logisch 0 entspricht einer unmaskierten Datenleitung. INITPM gibt z.B. das folgende Maskenbyte aus:

1 1 1 1 1 1 0 0

PIO-Port kontrolliert im genannten Beispiel nur die Leitungen PD0 und PD1 auf logisch 0.

SERVM: Diese Interrupt-Service-Routine hat folgende Funktionen:

a) Zwischenspeichern der CPU-Register-Inhalte.
b) Lesen der vier Eingabeleitungen des PIO-Datengatters Port D (PIO2, Port B).
c) Anzeige des Eingabe-Halbbytes (die vier niederwertigen Bits) auf den vier rechten 7-Segment-Displays. Die vier linken Anzeigen sind mit Nullen aufgefüllt. Diese Anzeige steht auf der Tastatur-/Anzeigeeinheit des Nanocomputers® ca. 5 Sekunden zur Verfügung.
d) Ausgabe des Eingabe-Halbbytes an die vier Ausgabeleitungen des PIO-Datengatters Port D. Das erfordert den Austausch der High- gegen die Low-Halbbytes.
e) Umspeichern des CPU-Zustandes.
f) Freigabe von maskierbaren CPU-Interrupts.
g) Rückgabe der Steuerung zur MAIN-Routine.

Bild 7-19 zeigt den Steuerungsablauf zwischen den Routinen INITPM, MAIN und SERVM.

Eine weitere Tatsache über den Zustand der Quittungsbetriebssignale beim PIO-Betrieb nach Methode 3 sollte man nicht außer acht lassen. Wenn Gatter C nach Methode 3 arbeitet, wird die CRDY-Leitung im Zustand low gehalten. Arbeitet Gatter D nach Methode 3, muß die DRDY-Leitung im Zustand low bleiben. Dies gilt nur dann nicht, wenn Port C nach der bidirektionalen Methode arbeitet, welche die Quittungsbetriebsleitungen von Port D benötigt. Im Versuch Nr. 5 haben Sie bereits das DRDY-Signal bei der Programmierung von Port D für den Betrieb nach Methode 3 benutzt.

2. Schritt

Bild 7-20 zeigt die Zeitfolge beim PIO-Betrieb nach Methode 3. Die CPU

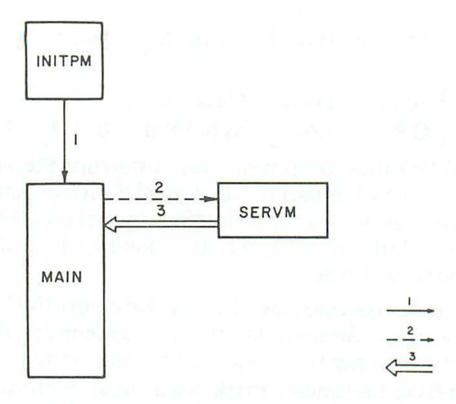


Bild 7-19. Steuerungsablauf der Routinen INITPM, MAIN und SERVM.
1 = Beendigung der INITPM-Routine
2 = Antwort auf Interrupt
3 = Rückkehr von Interrupt

kann eine normale Lese- oder Schreiboperation zu oder von einem PIO-Gatter nach Methode 3 ausführen. Die Zeitfolge für eine Leseoperation ist dieselbe wie für Methode 0. Bild 7-20 zeigt die Zeitfolge für eine Leseoperation. Die zur CPU zurückgegebenen Daten bestehen aus Ausgabe- und Eingabedaten von den jeweils gewählten Leitungen. Die Daten auf den Ausgabeleitungen werden vom Inhalt der letzten Byteausgabe bestimmt. Die Daten auf den Eingabeleitungen sind abhängig von dem Byte, das sich unmittelbar vor der ansteigenden Flanke von $\overline{RD}$ auf dem Daten-BUS befindet.

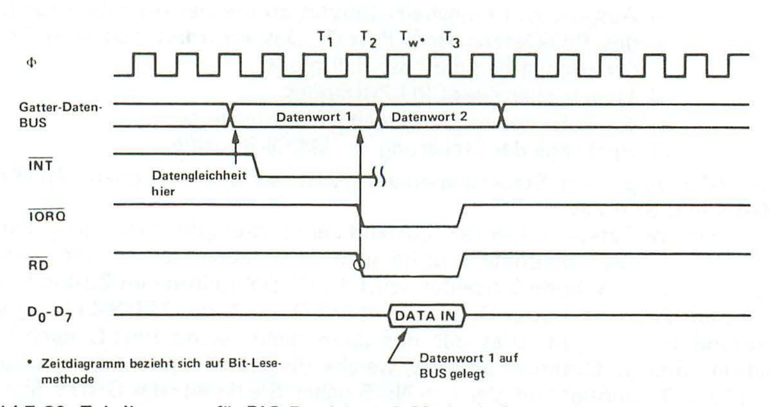


Bild 7-20. Zeitdiagramm für PIO-Betrieb nach Methode 3.

Der PIO kontrolliert die unmaskierten Datengatterleitungen ständig auf das Erscheinen eines Bit-Musters, das ein Interrupt-Signal erzeugen kann. Erscheint ein solches Bit-Muster, wird eine Unterbrechung ausgelöst. Bevor

eine weitere Unterbrechung folgen kann, müssen die Erzeugungsbedinungen für die letzte Unterbrechung erst gelöscht sein. Erst dann sind die Voraussetzungen für ein neues Interrupt-Signal geschaffen. Beispiel: Das PIO-IC ist so programmiert, daß ein Interrupt-Signal folgt, wenn eine Leitung logisch 0 ist. Wird währenddessen eine weitere Datenleitung aktiviert, hat dies kein weiteres Interrupt-Signal zur Folge. Bevor das PIO-IC eine weitere Unterbrechung erzeugt, müssen alle Datenleitungen zunächst einmal inaktiv (logisch 1) werden. Erst wenn eine der Datenleitungen erneut auf logisch 0 geht, löst dies ein neues Interrupt-Signal aus.

3. Schritt
Bauen Sie die Schaltung aus Bild 7-18 auf. Setzen Sie die Logik-Schalter SW0 und SW3 auf logisch 1. Starten Sie das Programm ab Speicherstelle INITPM. Setzen Sie den Schalter SW3 von logisch 1 auf logisch 0. Was passiert?
Die Anzeigen bleiben unverändert.

4. Schritt
Stellen Sie mit den Schaltern SW2 und SW3 alle möglichen Kombinationen von logischen Zuständen ein. Auch jetzt darf sich die Anzeige nicht verändern.

5. Schritt
Setzen Sie die Logik-Schalter SW1 . . . SW3 auf logisch 1 und SW0 auf logisch 0. Die LEDs LM4 . . . LM7 zeigen ein hexadezimales E und die Tastenfeld-/Anzeigeeinheit des Nanocomputers® 0E an. Die Interrupt-Service-Routine gibt die Steuerung an die MAIN-Routine zurück, obwohl SW0 im Zustand logisch 0 bleibt. Es entsteht so nur genau eine Unterbrechung. Es entsteht ein Interrupt-Signal, wenn auf den Datenleitungen die dazu erforderliche Information ansteht. Selbst wenn die Information nach Ausführung des Interrupts erhalten bleibt, ist dies ohne Bedeutung. Bevor ein weiteres Interrupt-Signal ausgelöst wird, muß auf den Datenleitungen erst eine Information anstehen, die keine Unterbrechung hervorruft.

6. Schritt
Wenn die Anzeige wieder die Bearbeitung der MAIN-Routine anzeigt, den Schalter SW1 von logisch 1 nach logisch 0 schalten.
Es entsteht kein Interrupt-Signal, weil die Bedingungen auf den Datenleitungen nicht stimmen.

7. Schritt
Setzen Sie die Logik-Schalter SW1 und SW0 zurück auf logisch 1. Die dadurch auf den Datenleitungen anstehende Information ist für ein Interrupt-Signal irrelevant. Setzen Sie nun SW1 wieder auf logisch 0.
Es wird ein Interrupt-Signal ausgelöst.

8. Schritt
Setzen Sie die vier Schalter SW0 . . . SW3 auf logisch 1. Lösen Sie mit SW0 ein Interrupt-Signal aus und bringen Sie anschließend den Schalter wieder

möglichst schnell auf logisch 1. Nach einigen Versuchen gelingt es Ihnen dadurch eventuell, alle Ausgabeanzeigen auf logisch 1 zu setzen. Das zeigt, wie lange die Interrupt-Service-Routine SERVM benötigt, um den CPU-Zustand zwischenzuspeichern (ca. 53 Mikrosekunden). Das entspricht in etwa der Zeitspanne zwischen der Interrupt-Erzeugung und der Aktivierung des $\overline{\text{RD}}$-Signals durch den IN-Befehl. Falls Sie festestellen wollen, welche Leitung das Interrupt-Signal erzeugt, müssen Sie das Signal mit Hilfe von SW0 auffangen.

9. Schritt
Ändern Sie an der Speicherstelle CWORD+1 in der Routine INITPM den Befehl 97 in D7. Dadurch nimmt das AND/OR-Bit im Interrupt-Steuerwort den Zustand logisch 1 an. Zur Interrupt-Erzeugung müssen nun SW0 und SW1 logisch 0 sein.

10. Schritt
Logik-Schalter SW0 . . . SW3 auf logisch 1 setzen. Das Programm bei INITPM starten. SW1 umschalten. Diese Änderung hat kein Interrupt-Signal zur Folge. Ebenfalls kein Interrupt-Signal erzeugt die Kombination SW1 = logisch 1 und SW0 = logisch 0. Erst wenn beide Schalter den Zustand logisch 0 an den Daten-BUS weitergeben, geht das Programm an die Interrupt-Service-Routine über.

11. Schritt
Ist die Unterbrechung bearbeitet, SW0 umschalten ("0" - "1" - "0"). Es beginnt eine weitere Interrupt-Bearbeitung. Durch das Umschalten von SW0 ist die Interrupt-Bedingung erfüllt: SW0 und SW1 müssen beide logisch 0 sein.

Anmerkung: Die Quittungsbetriebsleitungen $\overline{\text{STB}}$ und RDY sind in diesem Versuch ohne Bedeutung.

VERSUCH NR. 7

Der Versuch befaßt sich mit dem internen Prioritätsschema der PIO-ICs. Da beide PIO-Ports im Interrupt-Betrieb arbeiten, ist für eine zufriedenstellende Bearbeitung durch die CPU eine Rangfolge erforderlich.

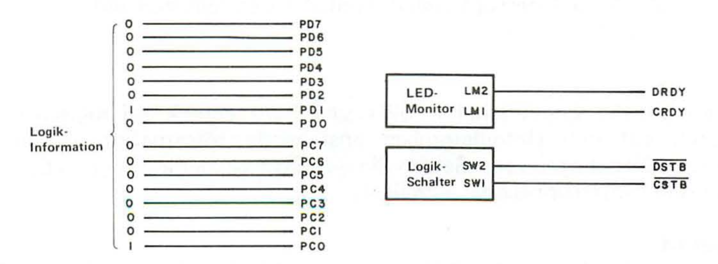


Bild 7-21.

Programme MAIN, SERVIC, SERVID und INITPP

Neben den Routinen MAIN, SERVIC und SERVID benutzt dieser Versuch noch die INITPP-Routine.

Objekt-Code		Quell-Code	Bemerkung
		NAME INITPP	
ED5E	INITPP:	IM2	; Interrupt-Modus 2
21000F		LD HL,TABLE	; Adresse der Vektortabelle
7C		LD A,H	; HI-Adreßbyte
ED47		LD I,A	; Interrupt-Vektor setzen
FD211904		LD IY,SERVIC	; Service für Port C
FD220A0F		LD (TABLE+0AH),IY	; Tabelle einsetzen
FD211F04		LD IY,SERVID	; Port D
FD22080F		LD (TABLE+08H),IY	; Tabelle einsetzen
3E0A		LD A,0AH	; Interrupt-Vektor für C setzen
D30A		OUT (0AH),A	
3E08		LD A,08H	; Interrupt-Vektor für D setzen
D30B		OUT (0BH),A	
08		EX AF,AF'	; Format für CONVDI setzen
3E40		LD A,40H	
08		EX AF,AF'	
3E4F		LD A,4FH	; Methode 1 für C und D
D30A		OUT (0AH),A	
D30B		OUT (0BH),A	
3E87		LD A,87H	; C und D freigeben
D30A		OUT (0AH),A	
D30B		OUT (0BH),A	
DB08		IN A,(08H)	; CRDY und DRDY
DB09		IN A,(09H)	; initialisieren
C3C302		JP MAIN	

1. Schritt

Bauen Sie für diesen Versuch die in Bild 7-21 dargestellte Schaltung auf. Da 16 Daten-BUS-Anschlüsse erforderlich sind, reichen die Logik-Schalter für das "Setzen der Logik" auf die BUS-Leitungen nicht aus. Deshalb werden die BUS-Leitungen direkt mit der +5 V- und der Masseschiene auf der SK-10-Platine verbunden. Wollen Sie z.B. eine logische 1 setzen, muß die entsprechende BUS-Leitung an +5 V angeschlossen werden; für 0 an Masse.

2. Schritt

Neben der MAIN-Routine werden in diesem Versuch die Initialisierungs-Routine INITPP und die beiden Interrupt-Service-Routinen SERVIC und SERVID benutzt. An dieser Stelle folgt deshalb die Beschreibung der neuen INITPP-Routine. Ebenfalls werden die bereits benutzten Routinen SERVID und SERVIC kurz beschrieben.

INITPP: PP steht für PIO-Priorität. Diese Initialisierungs-Routine hat folgende Funktionen:

a) Setzen der Z-80-Interrupt-Methode auf Modus 2.

b) Initialisierung von Register I.

c) Einrichten der Interrupt-Vektortabelle.
d) Laden von PIO-Port C und Port D mit den entsprechenden Interrupt-Vektoren.
e) Programmieren der Ports C und D für die Betriebsart nach Methode 1.
f) Bestimmung des Interrupt-Steuerworts für beide Ports mit freigebenen Unterbrechungen.
g) Initialisierung von CRDY und DRDY.

SERVIC und SERVID sind Interrupt-Service-Routinen mit folgenden Funktionen:

a) Zwischenspeichern der CPU-Registerinhalte.
b) Eingabe der Datenbytes von den PIO-Ports C bzw. D.
c) Anzeige des Eingabebytes für 10 Sekunden anstelle des Zählerinhaltes auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers®.
d) Umspeichern der CPU-Registerinhalte.
e) Rücksprung in die MAIN-Routine.

Wie bereits erwähnt, teilen sich SERVIC und SERVID das Codesegment SERVI. Den Steuerungsablauf zwischen MAIN, INITPP, SERVIC und SERVID zeigt Bild 7-22.

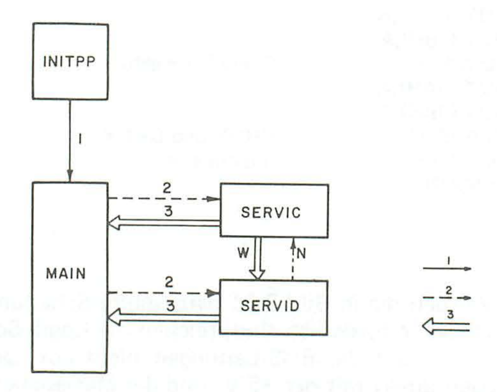


Bild 7-22. Steuerungsablauf der Routinen INITPP, MAIN, SERVIC und SERVID.
1 = Beendigung der INITPP-Routine
2 = Antwort auf Interrupt
3 = Rücksprung vom Interrupt

Ändern Sie im SERVI-Codesegment den NOP-Befehl (Speicherstelle ENABG) in einen EI-Befehl (hex FB). Diese Änderung gibt Interrupt für fast die gesamte Bearbeitungszeit aller Interrupt-Service-Routinen frei. Deshalb kann die Z-80-CPU eine Interrupt-Anforderung bestätigen, die bereits während der Bearbeitung einer Unterbrechung ankommt. Verzichtet die CPU auf die erste Interrupt-Bearbeitung zu Gunsten einer zweiten, spricht man von PREEMPTION. Durch diesen VORGRIFF wird die zweite Unterbrechung der ersten vorgezogen. Geben Interrupt-

Service-Routinen solche Unterbrechungen nicht wieder frei, handelt es sich um *nichtpreemptive* Bearbeitungsdisziplinen.

3. Schritt
Beginnen Sie die Programmausführung mit der INITPP-Routine. Erzeugen Sie mit SW1 ein Interrupt-Signal.
Die auf dem Daten-BUS von Port C anstehende Hexziffer 01 erscheint für ca. 10 Sekunden auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers®. Das CDRY-Signal wird aktiviert und bleibt logisch 1.

4. Schritt
Ist die Stack-Pointer-Adresse wieder 0F00, erzeugen Sie mit dem Schalter SW2 ein $\overline{DSTB}$-Signal. Das ist ein Interrupt-Signal für Port D.
Auf dem Daten-BUS für Port D steht die Hexziffer 02 zur Verfügung. Sie erscheint ebenfalls für ca. 10 Sekunden auf der Tastenfled-/Anzeigeeinheit des Nanocomputers®. Das Lesesignal DRDY geht auf logisch 1 und bleibt in diesem Zustand.

5. Schritt
Ist die letzte Interrupt-Bearbeitung beendet, Port C durch Schalten von SW1 abtasten. Während die SERVICE-Routine läuft, ebenfalls Port D durch Schalten von SW2 abtasten.
Sobald $\overline{DSTB}$ aktiviert ist, nimmt DRDY den Low-Zustand ein. Das PIO-IC speichert ein noch bevorstehendes Interrupt. Nachdem SERVIC beendet ist, wird die Steuerung der CPU als Antwort auf die noch ausstehende Unterbrechung von Port D zur SERVID-Routine übertragen.

6. Schritt
Ist die Adresse des Stack-Pointers wieder 0F00, $\overline{CSTB}$ – und während SERVIC noch läuft – ebenfalls $\overline{DSTB}$ aktivieren. Es erfolgt eine GESCHACHTELTE Interrupt-Behandlung. Sobald Port C abgetastet ist, erscheint das Eingabebyte 01 anstelle von 02 auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers®. Port C hat gegenüber Port D Vorrang. Daher verzichtet Port D zu Gunsten von Port C auf die sofortige Bearbeitung. Die umgekehrte Reihenfolge der geschachtelten Unterbrechung ist nicht möglich (siehe 6. Schritt). Fordert Port D während der Bearbeitungszeit von Port C eine Unterbrechung an, muß Port D warten, bis die Bearbeitung von Port C beendet ist. Umgekehrt unterbricht eine Bearbeitungsanforderung von Port C die momentale Bearbeitung von Port D, bis Port C bedient ist. Die Abkürzung für "C hat Priorität gegenüber D" lautet

$$C > D$$

7. Schritt
Zeigt der Stack-Pointer wieder die Adresse 0F00, Port D erneut abtasten. Während die Service-Routine SERVID läuft, Port D noch einmal ansprechen.
Die Bearbeitung des zweiten Interrupt-Signals beginnt erst nach Bearbeitung der ersten Unterbrechung, also

$$D \not> D$$

Das verdeutlicht die Funktion des Interrupt-Abfertigungsflipflops innerhalb des PIO-ICs. Die interne PIO-Schaltung für das Interrupt-Warte- und das Interrupt-Abfertigungsflipflop zeigt Bild 7-23. Diese Interrupt-Steuerlogik benutzt auch die Z-80-Zähler-Zeitgeberschaltung (CTC), auf die das letzte Kapitel näher eingeht.

VERSUCH NR. 8

Dieser Versuch verdeutlicht einen Interrupt-Prioritätsaufbau, der *Verkettung* genannt wird. Das PIO-IC bedient Verkettungen über die Leitungen IEI und IEO. PIO1 und PIO2 sind Verkettungen auf der PC-Platine des Nanocomputers®.

Neben der Schaltung aus Bild 7-21 wird für diesen Versuch die in Bild 7-24 gezeigte Schaltung benötigt sowie folgende Routinen: INITDC, INITPP MAIN, SERVIC, SERVID, SERVIE und SERVIF. Die Funktionen INITPP, MAIN, SERVIC und SERVID sind bereits beschrieben.

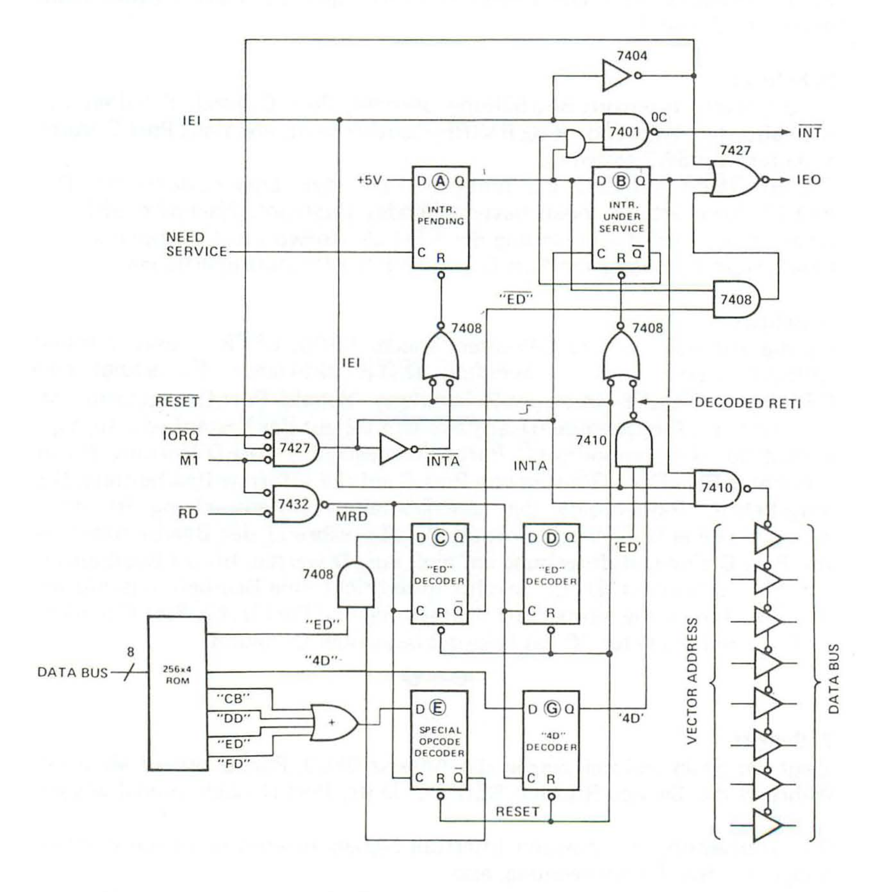


Bild 7-23. PIO-Interrupt-Steuerlogik.

Objekt-Code		Quell-Code	Bemerkung
		NAME INITDC	
ED5E	INITDC:	IM2	; Interrupt-Methode 2
21000F		LD HL,TABLE	; Adresse der Vektortabelle
7C		LD A,H	; HI-Adreßbyte
ED47		LD I,A	; Interrupt-Vektor setzen
FD212504		LD IY,SERVIE	; Service-Routine Port E
FD220E0F		LD (TABLE+0EH),IY	; Eingabe
			; Tabelle einsetzen
FD212B04		LD IY,SERVIF	; Service-Routine Port F
FD22100F		LD (TABLE+10H),IY	; Eingabe
			; Tabelle einsetzen
3E0E		LD A,0EH	; Interrupt-Vektor E laden
D30E		OUT (0EH),A	
3E10		LD A,10H	; Interrupt-Vektor F laden
D30F		OUT (0FH),A	
08		EX AF,AF'	; Format für CONVDI setzen
3E40		LD A,40H	
08		EX AF,AF'	
3E4F		LD A,4FH	; PIO-Methode 1 setzen
D30E		OUT (0EH),A	; Port E
D30F		OUT (0FH),A	; Port F
3E87		LD A,87H	; PIO freigeben
D30E		OUT (0EH),A	; Port E
D30F		OUT (0FH),A	; Port F
DB0C		IN A,(0CH)	; Initialisiere ERDY
DB0D		IN A,(0DH)	; Initialisiere FRDY
C37305		JP INITPP	

Objekt-Code		Quell-Code	Bemerkung
		NAME SERVIE	
C5	SERVIE:	PUSH BC	
0E0C		LD C,0CH	; Interrupt Port E
C33104		JP SERVI	

Objekt-Code		Quell-Code	Bemerkung
		NAME SERVIF	
C5	SERVIF:	PUSH BC	
0E0D		LD C,0DH	; Interrupt Port F
C33104		JP SERVI	

Erkennen Sie die Ähnlichkeit der Routine SERVIE und SERVIF mit SERVIC und SERVID? Diese Routinen unterscheiden sich nur durch die Gatteradresse an den Speicherstellen PORTC, PORTD, PORTE und PORTF. Im praktischen Betrieb ist es eine Verschwendung, vier Subroutinen dieser Größe aufzubewahren, die sich lediglich durch ein Byte unterscheiden. Sie dienen hier lediglich nur zur Demonstration.

1. Schritt

Im ersten Schritt machen Sie reichlichen Gebrauch von den Gattern A und B (auf der Platine des Nanocomputers® mit C und D bezeichnet) des PIO2 sowie von den Gattern A und B des PIO3. Die PIO3-Schaltung verbinden

Sie mit dem Anschluß SK-10 des Nanocomputers®. Innerhalb einer PIO-Schaltung hat Port A gegenüber Port B Vorrang (siehe Versuch Nr. 7). Wie ist der Prioritätsaufbau zwischen den ICs? Ein Prioritätsschema muß etwaige Probleme zwischen wetteifernden Bearbeitungsanforderungen lösen können. Dieser Versuch wird die Frage beantworten.

Bauen Sie zunächst die Schaltung aus Bild 7-24 auf. Der $\overline{CE}$-Eingang der PIO-Schaltung ist mit $\overline{IOQ3}$ verbunden. $\overline{IOQ3}$ dekodiert die Leitungen A7 bis A2 des Adreß-BUS und ist aktiv, wenn diese Leitungen folgendes Bitmuster aufweisen:

A7	A6	A5	A4	A3	A2
0	0	0	0	1	1

Da die Leitungen A1 und A0 mit den Anschlüssen C/D bzw. B/A verbunden sind, gilt für die I/O-Adressenleitungen für PIO3 folgende Port-Dekodierung:

Port	Adresse
Daten-Port A	0C
Daten-Port B	0D
Steuer-Port A	0E
Steuer-Port B	0F

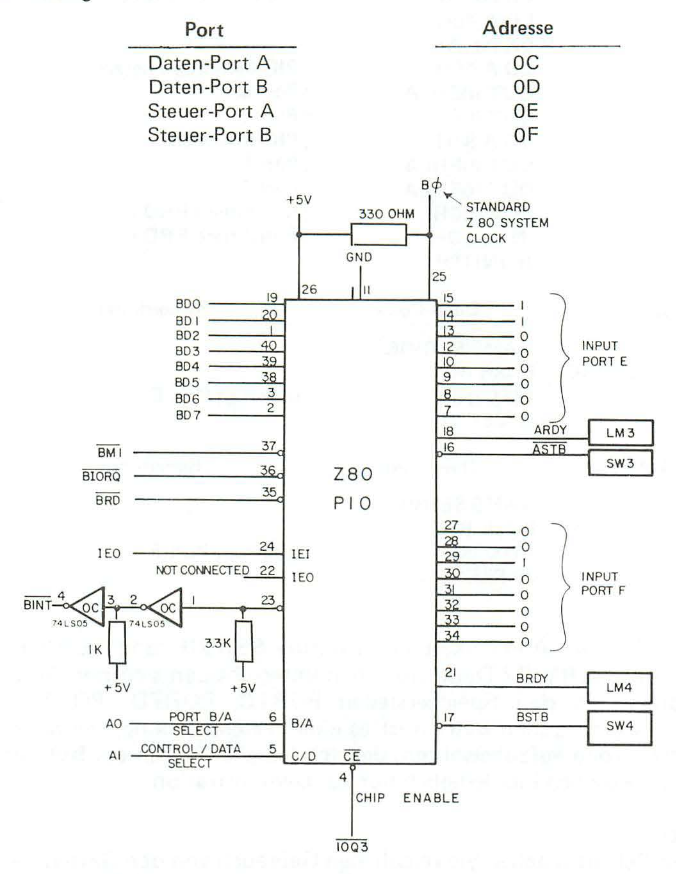


Bild 7-24. Anschlußbelegung für das externe PIO-IC.

Bild 7-25 zeigt die Verbindung zwischen dem externen PIO-IC3 und dem $\overline{\text{INT}}$-Eingang der CPU. In Bild 7-26 ist die Verbindung der drei PIO-ICs untereinander dargestellt. Dabei ist der IEO-Anschluß des vorhergehenden ICs mit dem IEI-Pin des folgenden ICs verbunden. Der IEI-Anschluß von PIO1 ist direkt an +5 V angeschlossen. Der IEO-Anschluß von PIO3 ist hingegen nicht beschaltet. Die Reihenschaltung der drei PIO-ICs nennt man *Verkettung*. Die Verkettung von sechs PIO-Ports ist ein Prioritätsaufbau mit folgenden Regeln:

- Jedes PIO-IC kann nur dann eine Unterbrechung anfordern, wenn der IEI-Anschluß logisch 1 (high) ist.
- Löst ein PIO ein Interrupt-Signal aus, geht der IEO-Anschluß des gleichen ICs auf logisch 0 (low).
- Ist der IEI-Eingang low, nimmt auch der IEO-Ausgang diesen Zustand an.

Diese Regeln haben folgende Konsequenzen:

1. PIO1 kann jederzeit eine Unterbrechung anfordern, da sein IEI-Eingang immer high ist.
2. Fordert PIO1 eine Unterbrechung an, geht der IEO-Ausgang in den Low-Zustand. Infolgedessen sind durch die Verkettung PIO2 und PIO3 für eine Interrupt-Anforderung gesperrt. Allgemein gilt: Fordert ein PIO-IC eine Unterbrechung an, sind durch die Verkettung alle folgenden PIO-ICs für Interrupt-Anforderungen gesperrt, da die IEI-Eingänge alle logisch 0 sind.
3. Für die insgesamt sechs PIO-Ports gilt folgendes Prioritätsschema: PIO1-Port A > PIO1-Port B > PIO2-Port A > PIO2-Port B > PIO3-Port A > PIO3-Port B.

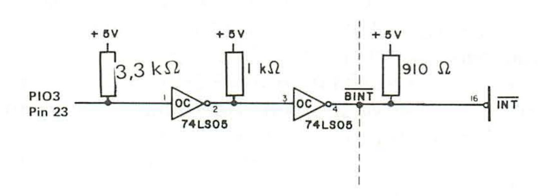


Bild 7-25. Anschluß der externen PIO-Schaltung an den $\overline{\text{INT}}$-Anschluß der Z-80-CPU.

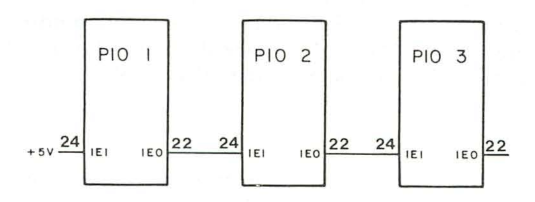


Bild 7-26. PIO-Verkettung.

In den folgenden Versuchsschritten wird diese Aussage experimentell bestätigt.

2. Versuch
Dieser Versuch benutzt die PIO-ICs 2 und 3. Die Routinen INITDC und INITPP programmieren alle vier Ports für die Betriebsart 1, der Eingabe-Methode. Die INITDC-Routine führt die gleiche Aufgabe für PIO3 durch wie INITPP für PIO2.
Belegen Sie die vier Port-Daten-BUS-Leitungen mit folgender Logik (die Belegung jeder einzelnen Leitung können Sie Bild 7-21 für Port C und D sowie Bild 7-24 für Port E und F entnehmen):

Port	Eingabebyte
PIO2 Datengatter A (Port C)	01
PIO2 Datengatter B (Port D)	02
PIO3 Datengatter A (Port E)	03
PIO3 Datengatter B (Port F)	04

Die vier Interrupt-Service-Routinen führen in bezug auf ihre Ports gleiche Funktionen aus. Die am deutlichsten sichtbare Funktion ist die Eingabe und Anzeige des Eingabebytes. Der Zusammenhang zwischen PIO-Port und Interrupt-Service-Routine ist:

Port	Interrupt-Service-Routine
PIO2 Datengatter A (Port C)	SERVIC
PIO2 Datengatter B (Port D)	SERVID
PIO3 Datengatter A (Port E)	SERVIE
PIO3 Datengatter B (Port F)	SERVIF

Prüfen Sie in der SERVI-Routine die Speicherstelle DSG + 6 (ENABG) auf ihren Inhalt. Sie sollte den EI-Befehl (hex FB) aufweisen. Dadurch gibt jede Interrupt-Service-Routine maskierbare Interrupts nach Zwischenspeichern der CPU-Registerinhalte frei. Bild 7-27 stellt die mögliche Steuerungsübertragungen zwischen den Interrupt-Service-Routinen und der MAIN-Routine dar. Die Ausführung von SERVID wird niemals zu Gunsten einer folgenden Routine verschoben. Allerdings unterbricht sie sofort zu Gunsten der SERVIC-Routine. Letztere wiederum hat gegenüber allen folgenden Routinen Vorrang. Für SERVIF verzichtet keine Routine auf die sofortige Bearbeitung.

3. Schritt
Aktivieren Sie Datengatter A (Port C) von PIO2 durch Schalten von SW1. Das Byte 01 erscheint für 10 Sekunden auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers®. Die neue Stack-Pointer-Adresse ist 0EEC. Das Interrupt-Flipflop IFF2 und somit auch IFF1 befinden sich im Zustand logisch 1.

4. Schritt
Wiederholen Sie den Ablauf aus Schritt 3 mit allen anderen PIO-Ports. Lassen Sie die jeweilige Interrupt-Bearbeitung erst ablaufen, bevor Sie das nächste Port prüfen. Bestätigen Sie mit diesem Schritt die Richtigkeit der Tabelle:

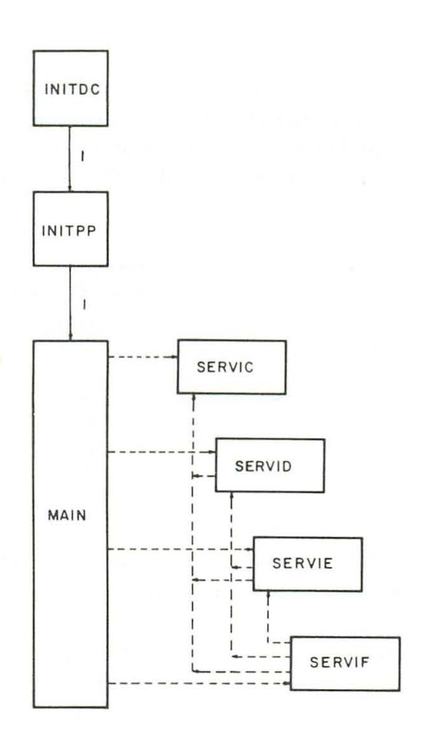


Bild 7-27. Steuerungsablauf zwischen den Routinen INITDC, INITPP, MAIN, SERVIC, SERVID, SERVIE und SERVIF.

Port	Anzeigebyte	Inhalt des Stack-Pointers	IFF1/IFF2
Port C	01	0EEC	1
Port D	02	0EEC	1
Port E	03	0EEC	1
Port F	04	0EEC	1

5. Schritt

Überprüfen Sie das Prioritätsverhältnis zwischen Port E und Port F von PIO3. Die höhere Priorität müßte Datengatter A (Port E) haben. Die Prüfung ist relativ einfach. Aktivieren Sie Port F. Bevor die dadurch ausgelöste Interrupt-Bearbeitung beendet ist, Port E (Datengatter A von PIO3) aktivieren. Es entsteht eine GESCHACHTELTE Interrupt-Bearbeitung. Aktivieren Sie die Datengatter in umgekehrter Reihenfolge, ist die Interrupt-Bearbeitung sequentiell, also nacheinander.

6. Schritt

Beweisen Sie die bereits aufgestellte Behauptung:

Datengatter B, PIO-Port D > Datengatter A, PIO-Port E

Gehen Sie wie in Schritt 5 vor.

7. Schritt

Betätigen Sie in schneller Folge die Schalter SW4, SW3, SW2 und SW1. Die Anzeige wechselt von 04 über 03, 02 nach 01, dem Datengatter mit der höchsten Priorität. Der Nanocomputer® bearbeitet die Routinen, bis vor dem Rücksprung zur MAIN-Routine wieder 04 erscheint. Nachstehend die entsprechende Tabelle:

Anzeige	Inhalt des Stack-Pointers
04	0EEC
03	0ED8
02	0EC4
01	0EB0
02	0EC4
03	0ED8
04	0EEC

8. Schritt

Betätigen Sie in der Reihenfolge die Schalter SW1, SW2, SW3 und SW4 schnell hintereinander. Prüfen Sie die Tabelle:

Anzeige	Inhalt des Stack-Pointers
01	0EEC
02	0EEC
03	0EEC
04	0EEC

9. Schritt

Überprüfen Sie weitere Abtastfolgen, damit Sie sich von der Gültigkeit folgender Verhältnisse überzeugen können:

PIO2-Port A > PIO2-Port B > PIO3-Port A > PIO3-Port B.

10. Schritt

Schalten Sie SW1 zweimal in schneller Folge.
Zwei Interrupts werden sequentiell (nacheinander) bearbeitet, weil ein Datengatter gegenüber sich selbst keine höhere Priotität besitzt. Die Stack-Pointer-Adresse springt also in jedem Fall immer erst nach 0EEC. Es folgt die Bearbeitung beider Interrupts, so daß die Anzeige ca. 20 Sekunden lang das Byte 01 zeigt.

Kapitel 8

TTL-IC-Tester

Das Kapitel beschreibt einen TTL-IC-Tester, mit dem man eine Vielzahl 14- und 16-poliger TTL-ICs prüfen kann. Der PI02 des Nanocomputers® ist Hauptbestandteil des Testers. Dabei sind beide I/O-Ports für die Betriebsart nach Methode 3 programmiert (Steuermethode). Mit Hilfe der Software kann man jede der 16 I/O-Leitungen als Ein- oder Ausgangsleitung benutzen. Dabei werden die 14 oder 16 Anschlußpins des zu prüfenden ICs mit den beiden Datengattern des PIO-ICs verbunden. CPU und PIO geben über ein relativ kurzes Programm Daten an das zu prüfende IC aus bzw. lesen Daten von dem IC ab. Die Daten werden dabei automatisch auf ihre Richtigkeit überprüft. Die Vielseitigkeit des PIO-ICs trägt wesentlich zur Leistung und Einfachheit des IC-Testers bei.

Das Kapitel vermittelt Ihnen folgende Kenntnisse:

- Einsatz des IC-Testers zur Funktionsprüfung verschiedener ICs der Serie 74LS.
- Verständnis des Verhältnisses zwischen den Bit-Positionen der PIO-Datengatter und der Pin-Konfiguration eines ICs mit 14 oder 16 Anschlüssen.
- Auswahl der Mindestzahl von Testworten aus insgesamt 65536 Möglichkeiten mit Hilfe der CHPTST-Routine.
- Wissen um das Zusammenarbeiten der Hard- und Software des IC-Testers. So können Sie feststellen, ob eine integrierte Schaltung defekt ist oder nicht.

Das Kapitel beschreibt die Betriebsprinzipien sowie die Soft- und Hardware des IC-Testers im Zusammenhang mit Low-Power-Schottky-TTL-ICs. Es stellt sich die Frage, ob ein solches Testgerät überhaupt erforderlich ist. Der Bau des Testers nimmt Zeit in Anspruch und benötigt einen Mikroprozessor. Hingegen sind die 74LS-ICs doch so billig, daß sich der ganze Aufwand nicht lohnt. In Großlabors ist der Einsatz eines IC-Testers unerläßlich, wenn es darum geht, unter tausend ICs die defekten herauszusuchen. Im Gegensatz zu den bekannten Testgeräten arbeitet der IC-Tester ohne den üblichen Kabelbaum, der die Verbindung zwischen dem zu testenden IC und dem Tester herstellt. Der nachfolgend beschriebene IC-Tester ist ein einfaches Prüfgerät, das in der Hauptsache aus dem PIO-IC und der Z-80-CPU besteht.

Das Grundprinzip des Testers ist relativ einfach. Die zu prüfende integrierte Schaltung ist ein Digitalsystem mit Mehrfach-Ein-/Ausgabe. Das IC benutzt eine Kombinationslogik, deren Pinanordnung im voraus bekannt sein muß. Unter *Kombinationslogik* versteht man ein IC, das mehrere logische Zustände kombiniert, um statisch-logische Werte an den

Ausgängen anzubieten. Typische Beispiele für Kombinationslogik sind die Standardgatter AND, NAND, OR und NOR. Durch den Ausdruck "Kombination" unterscheidet man die statisch-logischen Funktionen von den doch mehr dynamischen Eigenschaften anderer Logiksysteme, wie z.B. Flipflops und Zähler. Bei diesen Systemen sind die Übergänge zwischen beiden Logikpegel ebenso wichtig wie bestimmte statische Logikzustände. Zu einen späteren Zeitpunkt wird die Ausbau- und Anwendungsmöglichkeit des IC-Testers auch für diese Logiksysteme kurz erwähnt.
Der Prüfvorgang des in diesem Kapitel behandelten logischen Kombinationssystems besteht aus drei Stufen:

1. Erstellung der vollständigen Wahrheitstabelle für eine integrierte Bezugsschaltung, deren ordentliche Funktion vorausgesetzt wird.
2. Erstellen der vollständigen Wahrheitstabelle für den "Test" oder die "unbekannte" integrierte Schaltung.
3. Vergleich der beiden Wahrheitstabellen. Abweichungen weisen auf ein defektes IC hin.

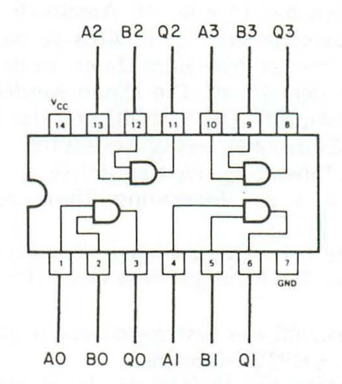


Bild 8-1. Vierfach-AND 74LS08 mit je zwei Eingängen. Die Wahrheitstabelle macht die Logikfunktion deutlich.

Falls Sie für jedes Gatter des ICs 74LS08 eine Wahrheitstabelle erstellen, erhalten Sie vier einzelne Tabellen. Sie können jedoch auch das IC in seiner Gesamtheit betrachten (Bild 8-2). In diesem Fall hat das IC insgesamt 8 Eingänge und vier Ausgänge. Es sind dabei 256 verschiedene Eingangskombinationen möglich. Diese globale Betrachtung ermöglicht dem IC-Tester eine bessere Beurteilung der Eigenschaften des zu prüfenden ICs.

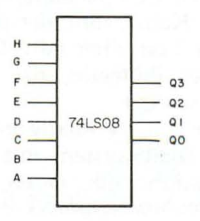


Bild 8-2. Alternative Betrachtungsweise des ICs 74LS08.

Die zum Prüfen des in Bild 8-2 dargestellten AND-Gattersysteme 74LS08 erforderlichen Funktionseinheiten sind aus Bild 8-3 ersichtlich.

Tabelle 8-1. Auszug aus der Wahrheitstabelle für die Betrachtungsweise nach Bild 8-1.

Inputs								Outputs			
H	G	F	E	D	C	B	A	Q0	Q1	Q2	Q3
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	1	0	1	1	1	0	0	0
0	0	0	0	1	1	0	0	0	1	0	0
0	0	0	0	1	1	0	1	0	1	0	0
0	0	0	0	1	1	1	0	0	1	0	0
0	0	0	0	1	1	1	1	1	1	0	0
				.				.			
				.				.			
				.				.			
				.				.			
1	1	1	1	1	1	1	1	1	1	1	1

Ein 8-Bit-Zähler belegt die acht Eingänge mit allen möglichen 256 Eingangskombinationen. Die Information der 4-Bit-Ausgänge wird in 256 aufeinanderfolgenden Speicherplätze eines Schreib/Lesespeichers (R/W Memory) gespeichert. Nachdem die vollständige Ausgangsinformation des Prüf-ICs gespeichert ist, vergleicht der IC-Tester diese mit der Information einer funktionsgerechten Schaltung.

Unabhängig von einigen Ausführungseinschränkungen kann der IC-Tester für alle ICs mit statischer Kombinationslogik benutzt werden. Dazu gehören z.B. Gatter, Puffer, Dekoder, Multiplexer, Addierer; im Prinzip also alle flankenfrei getriggerten ICs. Das Prüfungsprinzip: das IC erhält alle möglichen statischen Eingangskombinationen. Dabei tritt allerdings eine Unzulänglichkeit zutage. Die Eingabefolge ist immer dieselbe; mit jedem Schritt erhöht der Zähler um ein Bit. Obwohl dieses Prüfverfahren nur die "kriminellsten" Fehler nicht bemerkt, muß die Tatsache jedem Anwender bekannt sein.

Im Gegensatz zu den in Bild 8-3 dargestellten Funktionseinheiten wird bei dem in diesem Kapitel behandelten IC-Tester auf die Hardware und die Zählerlogik verzichtet. Was an Hardware noch übrig bleibt, sind der Speicher, die CPU, vier I/O-Ports des PIO-ICs und die Verbindungen zum Prüf-IC.

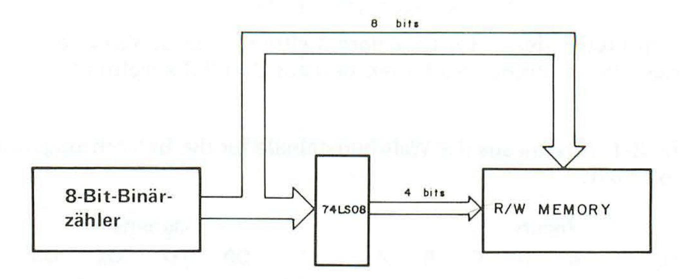


Bild 8-3. Funktionseinheiten eines IC-Testers.

Ferner sind noch zwei Maskenbytes erforderlich, eines für jeden nach der Steuermethode arbeitenden PIO-Port. Die Software realisiert den Bit-Zähler.

Folgende Informationen über das Prüf-IC müssen bekannt sein, damit der IC-Tester entsprechend gestaltet werden kann:

1. Der Versorgungsspannungsanschluß: Das IC muß mit einer geeigneten Versorgungsspannung verbunden sein.
2. Die Pin-Belegung, d.h. die Bedeutung aller Anschlußstifte.
3. Nichtbeschaltete Pins müssen entweder mit +5 V oder Masse verbunden sein (z.B. bei den ICs 74LS20 und 74LS30).
4. Das IC darf kein Analog/Digital-Baustein sein, das von externen oder internen RC-Zeitgeber-Schaltungen abhängig ist. Auch darf es über keine flankenaktive Eingänge verfügen. (z.B. die monostabilen Multivibratoren 74121, 74122, 74123 sowie das Flipflop 74LS74).

In Tabelle 8-2 sind die Ein- und Ausgabeeigenschaften verschiedener logischer Kombinationssysteme der TTL-Serie 74LS00 aufgeführt. Jedes aufgeführte IC hat entweder 14 oder 16 Anschlußstifte und kann mit Hilfe des hier beschriebenen Testers überprüft werden. Die noch zu beschreibende Software benötigt 4 K-Bytes vom Lese/Schreibspeicher, um ICs mit neun Eingangsanschlüssen prüfen zu können. Bei ICs mit mehr als neun Eingängen müssen die übrigen Pins entweder an +5 V oder an Masse angeschlossen sein. Um alle möglichen Eingangsinformationen zu erfassen, ist ein mehrstufiger Test erforderlich.

Es gibt mehrere Wege, um die Kapazität des IC-Testers für größere Logiksysteme auszubauen.:

1. Weitere Speicher hinzufügen.
2. Umschreiben der Software, um Bit-Muster einzugeben und resultierende Ausgaben in verschiedenen diskreten Blöcken zu vergleichen.
3. Umschreiben der Software, um einen byteweisen Vergleich zwischen den Ausgaben der Bezugs- und den unbekannten ICs durchzuführen.

Die letzte Alternative ist in bezug auf Speicherraum die zurückhaltenste, sie benötigt aber als zusätzliche Hardware ein zweites PIO-IC. Da zusätzlicher Speicherraum teuerer ist als ein zusätzliches PIO-IC, ist die 3. Alternative vorzuziehen.

Tabelle 8-2. Ein-/Ausgabeeigenschaften einiger logischer Kombinationssysteme bei ICs der Serie 74LSXX.

IC	Anzahl der Ausgabe-Bits vom Mikrocomputer zum IC	Anzahl der Eingabe-Bits vom IC zum Mikrocomputer	Anzahl der innerhalb des ICs nichtbelegten Anschlüsse
74LS02	8	4	0
74LS04	6	6	0
74LS05	6	6	0
74LS08	8	4	0
74LS30	8	1	3
74LS32	8	4	0
74LS42	4	10	0
74LS125	8	4	0
74LS139	6	8	0
74LS365	8	6	0
74LS54	10	1	1

INTERFACE-SCHALTUNG

Die einfachen Interface-Schaltungen für den IC-Tester des Nanocomputers® sind in den Bildern 8-4A und 8-4B dargestellt. Der PIO-2 auf der PC-Platine des Nanocomputers® ist mit dem Prüf-IC über die Anschlüsse der Verbindungseinheit C verbunden.

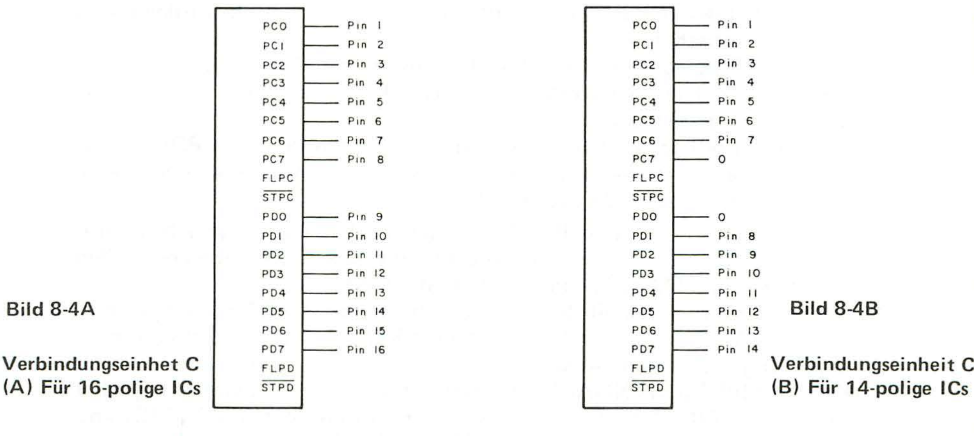


Bild 8-4A

Verbindungseinhet C
(A) Für 16-polige ICs

Bild 8-4B

Verbindungseinheit C
(B) Für 14-polige ICs

Bild 8-4. Schaltungsdiagramme für den IC-Tester.

Bei einem 16-poligen Prüf-IC sind alle Anschlüsse der PIO-Ports C und D mit dem IC verbunden. Sie dienen sowohl für die Prüf- als auch für die

Bezugsinformation. Anders verhält es sich bei einem 14-poligen Prüf-IC; es wird nur mit den Anschlüssen 1 . . . 7 und 10 . . . 16 vom PIO verbunden. Bild 8-4 zeigt die Anschlußbelegung für beide Fälle. Im ersten Falle sind alle Portanschlüsse PC0 . . . PC7 und PD0 . . . PD7 belegt, während im zweiten Fall die Leitungen PC7 und PD0 frei bleiben. Damit der Tester richtig funktioniert, müssen sie mit logisch 0 (Masse) verbunden sein.

Port C und Port D arbeiten nach der Steuermethode (Modus 3). Somit kann jede der 16-Daten-Portleitungen über ein I/O-Steuerbyte entweder als Eingang oder Ausgang geschaltet sein. Wie bereits erwähnt, haben die gesetzten Bits im PIO-I/O-Steuerbyte folgende Bedeutung:

Bit Dn = 0 im I/O-Steuerbyte von Port C bedeutet: Leitung PCn ist eine Ausgangsleitung.

Bit Dn = 1 im I/O-Steuerbyte von Port C bedeutet: Leitung PCn ist eine Eingangsleitung.

Bit Dn = 0 im I/O-Steuerbyte von Port D bedeutet: Leitung PDn ist eine Ausgangsleitung.

Bit Dn = 1 im I/O-Steuerbyte von Port D bedeutet: Leitung PDn ist eine Eingangsleitung.

Das I/O-Steuerbyte ist ein Parameter, den der Benutzer für den IC-Tester festlegt, um die wichtigen Eigenschaften des Prüf-IC zu beschreiben. Folgendes Verfahren ist notwendig, um ein Steuerbyte mit den Eigenschaften eines ICs in Zusammenhang zu bringen:

1. Den Versorgungsspannungsanschluß des ICs betrachtet der Mikrocomputer als Eingang. Daher müssen die entsprechenden PIO-Portleitungen im I/O-Steuerbyte den Wert logisch 1 erhalten. Die Anschlußstifte am IC selbst müssen mit +5 V und Masse verbunden sein.
2. Die Eingänge des Prüf-ICs sind für den Mikrocomputer Ausgänge, so daß den korrespondierenden PIO-Portleitungen der Wert logisch 0 zugeteilt wird.
3. Die Prüf-IC-Ausgänge stellen für den Mikrocomputer Eingänge dar. Die entsprechenden Bit-Positionen des Steuerbytes erhalten infolgedessen den Wert logisch 1.
4. Intern nichtbelegte Pins des Prüf-ICs müssen mit Masse verbunden sein. Ihre Bits im I/O-Steuerbyte sind logisch 1, da der Mikrocomputer sie als Eingang betrachtet.
5. Für ein 14-poliges Prüf-IC werden die Positionen PC7 und PD0 jeweils als Eingang angesehen und erhalten somit in ihren entsprechenden I/O-Steuerbytes den Wert logisch 1.

Nachdem die zu setzenden Bits für die beiden I/O-Steuerbytes bestimmt sind, leitet der Benutzer diese Information über zwei Speicherstellen (MASKW + 1 und MASKW) an den IC-Tester weiter.

Der Benutzer lädt manuell das I/O-Steuerbyte und Port C in die Speicherstelle MASKW und das für Port D in MASKW + 1. Daher wird eine PIO-Datenleitung, die dem Ausgangspin eines Prüf-ICs entspricht, für die MIKROCOMPUTEREINGABE benutzt und in den Zustand logisch 1 versetzt. Eine PIO-Datenleitung, die dem Eingangspin eines Prüf-ICs entspricht, wird für die MIKROCOMPUTERAUSGABE benutzt und geht auf logisch 0. Die Anschlüsse der Versorgungsspannung und die nichtbelegten Pins des Prüf-ICs sind für den Mikrocomputer Eingänge, weil die logischen Zustände während des Tests konstant bleiben. Anhand der Ein- und

Ausgabebits des Mikrocomputers erstellt die CHPTST-Routine die Wahrheitstabelle für das Prüf-IC. Bild 8-5 zeigt als Beispiel die Ermittlung des Maskenworts für das IC 74LS08. Tabelle 8-3 gibt für mehrere logische Kombinationssysteme der Serie 74LSXX die LO- und HI-Adreß-Bytes an.

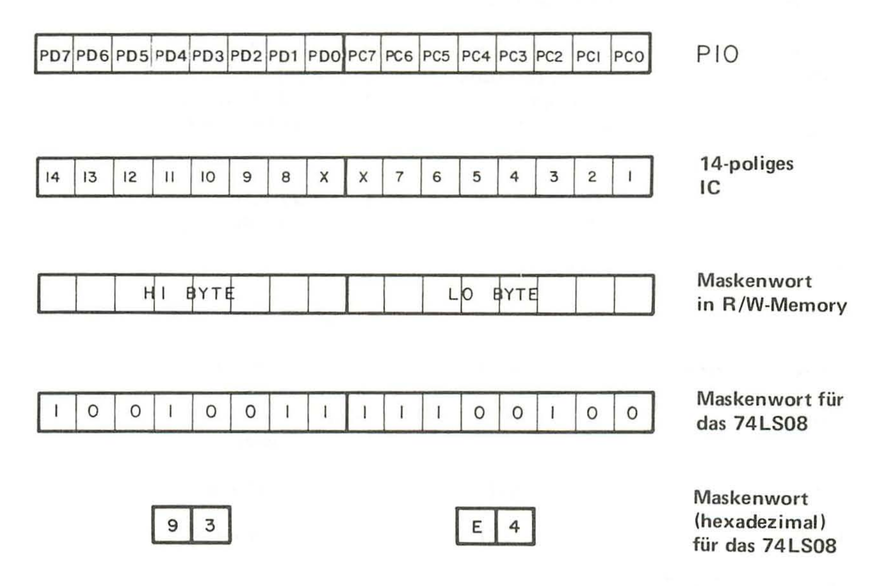


Bild 8-5. Steuerbyte-Aufbau für das vierfache AND-Gatter IC 74LS08.

SOFTWARE

Dieser Abschnitt beschreibt in kurzen Zügen die Software für den IC-Tester. Es folgt das Listing der CHPTST-Routine. Die absolute Adresse der Speicherstelle für diese Routine entnehmen Sie der Master-Symbol-Tabelle im Anhang A.

Tabelle 8-3. Steuerbytes mehrerer logischer Kombinationssysteme.

IC	Maskenwort	
	HI-Byte	LO-Byte
74LS02	C9	C9
74LS04	AB	EA
74LS05	AB	EA
74LS08	93	E4
74LS30	CF	C0
74LS32	93	E4
74LS42	87	FF
74LS125	93	E4
74LS139	8F	F8
74LS365	95	D4

Objekt-Code		Quell-Code	Bemerkung
		NAME CHPTST	
3E03	CHPTST:	LD A,03H	; PIO-Interrupt rück- ; setzen; Flipflop freigeben
D30A		OUT(0AH),A	
D30B		OUT(0BH),A	
2A0300		LD HL,(MASKW)	; Maskenwort für Schal-
010AFF		LD BC,0FF0AH	; tung setzen
ED41		OUT (C),B	; PIO-Port A, Modus 3
ED69		OUT (C),L	; I/O-Maske für Port A
0C		INC C	; Port B verändern
ED41		OUT (C),B	; PIO-Port B, Modus 3
ED61		OUT (C),H	; I/O-Maske für Port B
	;		
31A00F	REF:	LD SP,CHPSTK	; Stackpointer initiali- ; sieren
DD210008		LD IX,REFIC	; Zeiger auf Referenz- ; IC initialisieren
010000		LD BC,0000H	; Zählerwort initiali- ; sieren
CD8806		CALL STORE	; Bezugstabelle erstel- ; len
00	ENDREF:	NOP	
	;		
31A00F	UNKN:	LD SP,CHPSTK	; Stackpointer initiali- ; sieren
DD21000C		LD IX,UNKIC	; Zeiger auf Prüf-IC ; initialisieren
010000		LD BC,0000H	; Zählerwort initiali- ; sieren
CD8806		CALL STORE	; Ausgangstabelle des ; Prüf-ICs erstellen
210008	COMPAR:	LD HL,REFIC	; unter Verwendung des ; CPI-Befehls für den ; Vergleich die beiden ; Tabellen vorbereiten
11000C		LD DE,UNKIC	; HL zeigt auf Refe- ; renz-Tabelle, DE ; zeigt auf Tabelle ; des Prüf-ICs
1A	NEXTB:	LD A,(DE)	; Byte des Prüf-ICs ; in Akku laden
EDA1		CPI	; mit (HL) vergleichen
2037		JR NZ,BAD	; Falls die Tabellen ; voneinander abwei- ; chen, ist Prüf-IC ; defekt
13		INC DE	; Falls Tabellen gleich ; zum nächsten Byte- ; Test übergehen

```
EA7D06              JP PE,NEXTB       ;Wenn P/V-Flag = 1,
                                      ;nächstes Byte testen
1833      GOOD:     JR START          ;Ist P/V-Flag = 0,
                                      ;ist auch Zählerinhalt
                                      ;BC Null, alle Bytes
                                      ;sind getestet
110000    STORE:    LD DE,0000H       ;Initialisiere Test-
                                      ;wort
2A0300    NTEST:    LD HL,(MASKW)     ;HL mit Maskenwort
                                      ;laden
7B                  LD A,E            ;16-Bit-AND-Maske und
                                      ;Testwort ausführen

A5                  AND L
6F                  LD L,A
7A                  LD A,D
A4                  AND H
67                  LD H,A
7C        MASK:     LD A,H            ;Prüfen, ob Ergebnis des
                                      ;16-Bit-AND = 0 ist
B5                  OR L
201B                JR NZ,NXTWD       ;falls nicht 0, zum
                                      ;nächsten Byte übergehen
          ;
7B        TEST:     LD A,E            ;falls 0, ist es ein
                                      ;gültiges Testwort

D308                OUT (08H),A
7A                  LD A,D
D309                OUT (09H),A
2A0300              LD HL,(MASKW)     ;Maskenwort für IC nehmen
DB08                IN A,(08H)        ;LO-Byte von IC eingeben
A5                  AND L             ;LO-Byte maskieren
DD7700              LD (IX),A         ;LO-Byte speichern
DD23                INC IX            ;IX aktualisieren
DB09                IN A,(09H)        ;HI-Byte von IC eingeben
A4                  AND H             ;HI-Byte maskieren
DD7700              LD (IX),A         ;HI-Byte speichern
DD23                INC IX            ;IX aktualisieren
03                  INC BC            ;Zähler um 2 erhöhen

03                  INC BC
          ;
13        NXTWD:    INC DE            ;nächstes Testwort
                                      ;nehmen
7A                  LD A,D
B3                  OR E
20D3                JR NZ,NTEST       ;Falls DE nicht 0 ist,
                                      ;zurück, um nächstes
                                      ;Testwort zu prüfen
C9                  RET               ;Ist DE = 0, hat das
                                      ;Programm die komplette
                                      ;Ausgangstabelle erstellt

1800      BAD:      JR START          ;IC ist defekt, zu-
                                      ;rück zu START
```

```
18AD    START:   JR UNKN          ; Springe zur Test-
                                  ; routine für Prüf-
                                  ; IC
```

Nachstehend eine Erörterung des ausführbaren Codes, der Datentabellen und Datenpuffer für diese Routine:

Speicher-Bezeichnung	*Beschreibung*
MASKW bis MASKW + 1	LO-Steuer- oder Maskenbyte an der Speicherstelle MASKW, gefolgt von dem HI-Steuer- oder Maskenbyte an der Speicherstelle MASKW + 1 für das Prüf-IC.
CHPTST	Der ausführbare Code für den TTL-IC-Tester.
REFIC bis REFIC + 03FFH	Lese/Schreib-Speicherplatz für die Ausgabebits von der verallgemeinerten Wahrheitstabelle für die integrierte Bezugsschaltung. (Ein Beispiel ist in Tabelle 8-1 gezeigt). Die in diesem Raum gespeicherte Tabelle nennt man BEZUGS-Antwort-Tabelle.
UNKIC bis UNKIC + 03FFH	Lese/Schreib-Speicherplatz für die Ausgabebits von der verallgemeinerten Wahrheitstabelle für die unbekannte integrierte Schaltung (ein Beispiel ist in Tabelle 8-1 gezeigt). Die in diesem Raum gespeicherte Tabelle ist die UNBEKANNTE Antwort-Tabelle.
CHPSTK	Der Anfangswert für den Stackpointer (Stapelzeiger).

CHPTST beginnt mit der Initialisierung der Ports C und D des PIO2. Die Initialisierung der I/O-Bits in beiden Ports hängt von der Art des I/O-Aufbau-Steuerwortes ab, das der Anwender in das Registerpaar HL geladen hat. In diesem Steuerwort bedeutet eine logische 0, das entsprechende Bit im PIO-Port ist ein Ausgabebit (Ausgabe vom PIO zum Prüf-IC). Eine logische 1 bedeutet, das entsprechende Bit im PIO-Port ist ein Eingabebit (Eingabe von dem Prüf-IC zum PIO). Port C entspricht dem LO-Maskenbyte und Port D dem HI-Maskenbyte.
Bei den Routinen REF und UNKN handelt es sich um Register-Initialisierungsroutinen für die Bezugs- und Prüf-ICs. Diese Routinen initialisieren den Stackpointer, ein Hinweis-Register. Es definiert a) den Speicherbereich für die Antwort-Tabelle und b) das BC-Registerpaar als Zähler für die in der Antwort-Tabelle gespeicherte Byteanzahl. Die Routine COMPAR führt einen byteweisen Vergleich der Antwort-Tabellen für die Bezugs- und Prüf-ICs durch. Das im Registerpaar BC gespeicherte Zählerwort bestimmt die Anzahl der zu vergleichenden Bytes. Haltepunkte sind an den Speicherstellen GOOD und BAD gesetzt, um zwischen guten und schlechten (defekten) ICs zu unterscheiden. Am Ende des Vergleichs erscheint eine dieser beiden Haltepunkt-Speicherstellen auf der Anzeige des Nanocomputers®. Am Haltepunkt kann man die Prüf-ICs an den IC-Tester anschließen (Austauschen der ICs auf der Experimentierplatine).
Die wichtigste Software ist die STORE-Subroutine, welche die Masken-

routine einschließt, die ungültige Testwörter eliminiert. Es werden vier 16-Bit-Register benutzt:

Das BC-Registerpaar

Es zählt die Anzahl der Bytes, die in der Antwort-Tabelle für ein Prüf-IC gespeichert sind.

Das DE-Registerpaar

Das Prüfwort zählt von 0000 . . . FFFF. Eine AND-Operation im Programm bestimmt, welches der 65536 Prüfwörter zum Prüf-IC gelangt.

Das HL-Registerpaar

Es enthält das I/O-Steuerwort für das Prüf-IC. Logisch 0 definiert ein Ausgabebit vom Mikrocomputer und logisch 1 ein Eingabebit zum Mikrocomputer.

Das Indexregister IX

Der Registerinhalt bestimmt die Speicherstelle, in die das nächste Byte aus der Antwort-Tabelle des Prüf-ICs zu speichern ist.

Die Subroutine STORE initialisiert das Prüfwort auf 0000, lädt das Steuerwort und führt eine 16-Bit-AND-Operation zwischen den Prüf- und Steuerwörtern durch. Die MASK-Routine bestimmt, ob alle 16 Bits im Ergebnis der AND-Operation im Zustand logisch 0 sind oder nicht. Lautet die Antwort ja, ist das im Registerpaar DE gespeicherte Prüfwort ein gültiges 16-Bit-Wort. Es wird an Port C und D des PIO2 ausgegeben. Nach der Ausgabeoperation gehen zwei aufeinanderfolgende Bytes über den PIO an den Akkumulator, werden maskiert und dann in der Antwort-Tabelle gespeichert. Das im Indexregister IX gespeicherte Hinweiswort hat sich während des Vorgangs zweimal erhöht. Ebenfalls erhöht hat sich das Prüfwort. Hat es den Wert 0000 erreicht, ist der Testvorgang beendet. Falls das Prüfwort nicht 0 ist, wird das Prüfprogramm durch Rücksetzen zur Speicherstelle NTEST fortgesetzt.

Die Bedeutung des Prüfworts in der Operation der MASK-Routine läßt sich am besten anhand eines Beispiels erklären; z.B. anhand des ICs 74LS08 mit einem Steuerwort von 10010011 11100100 oder im Hexadezimalcode von 93 E4. Tabelle 8-4 zeigt eine Serie von Prüfwörtern zwischen 0000H und 0008H und das Ergebnis der entsprechenden AND-Operationen zwischen diesen Prüfwörtern und dem Steuerwort. Nur AND-Funktionen mit dem Ergebnis 00H, sind gültige Prüfwörter. Sie werden an das IC 74LS08 weitergegeben; in Tabelle 8-4 sind dies die ersten vier Wörter. Das Prüfwort ändert nur die Bits D0 und D1. Es handelt sich dabei um die Eingänge A0 und B0 des ersten AND-Gatters (siehe Bild 8-1). Die nächsten vier Prüfwörter sind ungültig, weil Bit D2 logisch 1 ist. Sie werden eliminiert, da es sich nicht um gültige Ausgabebits vom Mikrocomputer handelt. Das geht bereits aus dem Maskenwort bzw. dem Ergebnis der AND-Funktion in Tabelle 8-4 hervor. Jedes Prüfwort, dessen Bit D2 logisch 1 ist, hat auf den Test keinen Einfluß und kann also entfallen. Das gleiche gilt für die Bits D5, D6, D7, D8, D9, D12 und D15. Führt ein Prüfwort in einer der genannten Bit-Positionen logisch 1, ist es für den Test irrelevant. Auf diese Weise wählt der Nanocomputer® aus den 65536 möglich Prüfwörtern die für das IC 74LS08 256 relevanten aus. Sie entsprechen den 256 möglichen Eingangskombinationen der acht Eingänge des ICs.

Tabelle 8-4. Beispiel einer AND-Operation zwischen Prüf- und Maskenwörtern.

Prüfwort	Maskenwort	AND-Ergebnis	Prüfwort gültig?
00000000 00000000	10010011 11100100	00000000 00000000	ja
00000000 00000001	10010011 11100100	00000000 00000000	ja
00000000 00000010	10010011 11100100	00000000 00000000	ja
00000000 00000011	10010011 11100100	00000000 00000000	ja
00000000 00000100	10010011 11100100	00000000 00000100	nein
00000000 00000101	10010011 11100100	00000000 00000100	nein
00000000 00000110	10010011 11100100	00000000 00000100	nein
00000000 00000111	10010011 11100100	00000000 00000100	nein
00000000 00001000	10010011 11100100	00000000 00000000	ja

Einführung in die Versuche

Die folgenden Versuche machen Sie mit der Interface-Schaltung des TTL-IC-Testers sowie dem Programm vertraut. Im Versuch Nr. 1 bauen Sie die Interface-Schaltung auf und prüfen zwei ICs (74LS08 und 74LS32); beide haben ein identisches I/O-Steuerwort. In dem Versuch Nr. 2 stellen Sie den Maximalwert der gespeicherten Bytes in der Bezugs-Antwort-Tabelle des Prüf-ICs fest. Schließlich überprüfen Sie im dritten Versuch die Bezugs-Antowort-Tabelle für ein Gatter des ICs 74LS08.

VERSUCH NR. 1

1. Schritt
Programm CHPTST laden.

2. Schritt
Laden Sie das Maskenwort in den Lese/Schreibspeicher. Die absolute Adresse für die Speicherstelle MASKW ist 0003 und für MASKW + 1 ist 0004 (siehe auch Anhang A, Tabelle A-1).

Speicheradresse	Inhalt	Beschreibung
MASKW	E4	LO-Maskenbyte für 74LS08
MASKW + 1	93	HI-Maskenbyte für 74LS08

3. Schritt
BRK-Taste drücken, um die Haltepunktmethode (Breakpoint-Modus) einzuleiten, dann mit Hilfe der INC-Taste die folgenden Breakpoint-Adressen eingeben:

Speicherstelle	Bemerkung
ENDREF	Breakpoint für Beendigung der REF-Routine
GOOD	Breakpoint für "gute" ICs
BAD	Breakpoint für "schlechte" ICs

Die Reihenfolge ist bei der Eingabe der einzelnen Breakpoints ohne Bedeutung. Um den Breakpoint-Modus zu verlassen, drücken Sie erneut die BRK-Taste.

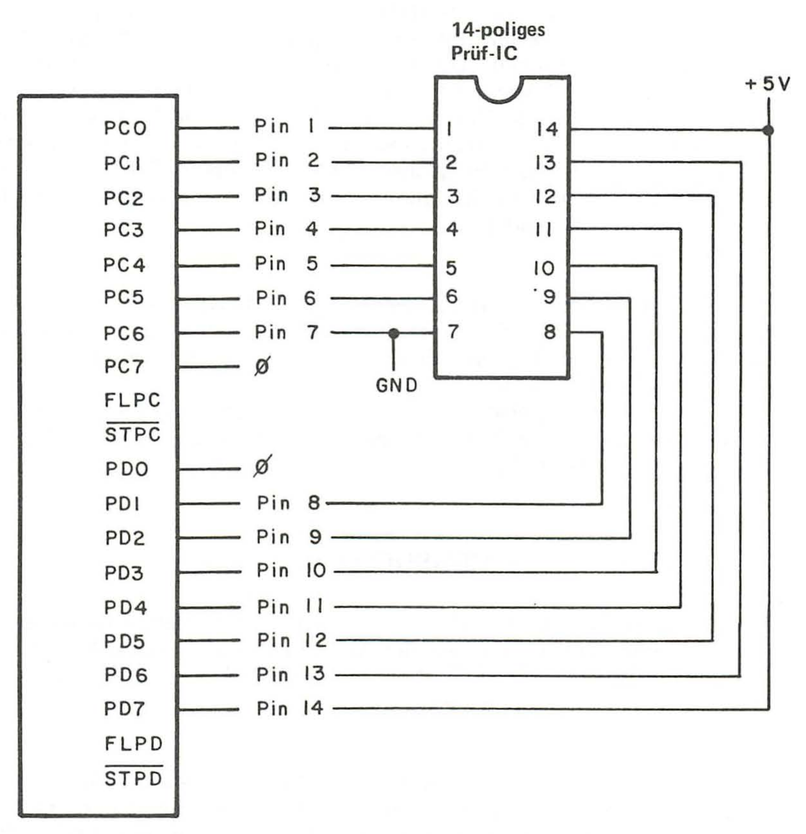


Bild 8-6. Schematische Darstellung der Schaltung für Versuch Nr. 1.

4. Schritt

Bezugs-IC 74LS08 in Prüf-Stellung bringen (Bild 8-6) und mit Ausführung des Programms CHPTST beginnen. Nach kurzer Zeit erscheint auf der Tastenfeldanzeige die Breakpointadresse ENDREF.

Der Nanocomputer® hat eine BEZUGS-Antwort-Tabelle erstellt. Unterbrechen Sie die Stromversorgung zum IC 74LS08 und nehmen das IC aus der Prüfposition heraus.

5. Schritt

Bringen Sie das zu prüfende IC (74LS32) in die Testposition und schließen

es an die Stromversorgung an. Die ICs 74LS08 und 74LS32 haben dasselbe Steuerwort. Der IC-Test beginnt ab der Routine UNKN, sobald Sie die GO-Taste drücken.
Das Programm endet bei dem Breakpoint BAD; auf der Anzeige erscheint die entsprechende Adresse. Es geht daraus hervor, daß die Bezugs-Antwort-Tabelle (Referenz-Tabelle) und die Ausgangstabelle des zu prüfenden ICs nicht übereinstimmen. Dafür können zwei Gründe verantwortlich zeichnen. Das Prüf-IC kann defekt oder ein falscher Typ sein. In diesem Beispiel trifft der zweite Grund zu. Das Bezugs-IC 74LS08 ist ein vierfaches AND-Gatter mit je zwei Eingängen, das Prüf-IC ist ein vierfaches ODER-Gatter mit je zwei Eingängen. Sind im anderen Fall Bezugs- und Prüf-IC vom gleichen Typ, ist letzteres defekt, wenn das Programm mit dem Breakpoint BAD endet.

6. Schritt.
Stromversorgung vom IC 74LS32 trennen und das IC aus der Prüfposition herausnehmen. IC 74LS08 in Prüfposition bringen und mit der Versorgungsspannung verbinden. Erneut die Test-Routine durch Drücken der GO-Taste starten.
Ist das IC in Ordnung, erscheint auf der Tastenfeldanzeige die Breakpointadresse von GOOD. Die Antwort-Tabelle des Prüf-ICs ist mit der Bezugs-Antwort-Tabelle identisch.

VERSUCH NR. 2

1. Schritt
Dieser Versuch arbeitet ohne Prüf-IC. Sie laden verschiedene Maskenwörter in den Speicher und bestimmen den Wert des Registerpaares BC, sobald die Programmausführung am Haltepunkt ENDREF stoppt. Der Versuch startet ab Speicherstelle CHPTST.

2. Schritt
Laden Sie das Maskenwort 93E4 in die Speicherstellen MASKW und MASKW + 1. Beachten Sie die richtige Reihenfolge; zuerst das LO-Byte E4, dann das HI-Byte 93. Starten Sie das Prüfprogramm ab CHPTST. Das Programm stoppt an Breakpoint ENDREF und zeigt die entsprechende Adresse auf dem Display.

3. Schritt
Selektor-LED in Position BC bringen. Welcher Wert zeigt das Display?
Der Inhalt des BC-Registerpaares ist 0200. Das entspricht 512 Bytes in der Bezugs-Antwort-Tabelle für das Maskenwort 93E4. Die Erklärung entnehmen Sie dem 4. Schritt.

4. Schritt
Wieviele Bytes sind in der Bezugs-Antwort-Tabelle für das IC 74LS08 gespeichert, dessen Maskenwort 93E4 ist?
Die binäre Form des Maskenwortes lautet

1001 0011 1110 0100

In diesem Wort sind 8 Bits logisch 0, das bedeutet insgesamt 256 Eintragungen in die Wahrheitstabelle. Für jeden Eintrag in die Wahrheitstabelle werden zwei Bytes gespeichert, ein LO-Byte (Gatter C) und ein HI-Byte (Gatter D). Das entspricht einer Gesamtzahl von 512 Bytes und der Anzeige von 0200 am Breakpoint ENDREF.

5. Schritt
Laden Sie in die Speicherstellen MASKW und MASKW + 1 das Maskenwort C9C9; es entspricht dem IC 74LS02. Starten Sie das Programm bei CHPTST. Sobald die Breakpointadresse ENDREF erscheint, prüfen Sie den Inhalt des Registerpaares BC. Stimmt der Inhalt mit Ihren theoretischen Betrachtungen überein? Der angezeigte Wert ist 0200, was wiederum 512 Bytes entspricht.

6. Schritt.
Wiederholen Sie den 5. Schritt mit dem Maskenwort CFC0 (74LS30). Welcher Registerinhalt hat das BC-Register am Breakpoint ENDREF? Es erscheint 0200. Das IC 74LS30 ist ein NAND-Gatter mit 8 Eingängen, also auch 8 Eingangspins. Außerdem hat das IC noch zwei Pins für die Versorgungsspannung und 3 unbelegte Pins. Die letzteren müssen beim IC mit logisch 0 (Masse) verbunden sein. Im I/O-Steuerwort sind die entsprechenden Bits logisch 1. Die acht Eingänge können insgesamt 256 verschiedene Eingangszustände annehmen, so daß die Bezugs-Antwort-Tabelle 512 Bytes speichert.

7. Schritt
Wieviele Bytes werden für den Dual-1-zu-4-Dekoder/Demultiplexer 74LS139 gespeichert? Der Dekoder hat sechs verschiedene Eingänge, (64 Eingangsmöglichkeiten) was 128 Bytes in der Antwort-Tabelle (hex 0080) entspricht.

8. Schritt
Entnehmen Sie aus der Tabelle 8-2 die Eigenschaften des ICs 74LS42. Wieviele Speicherstellen benötigt die Bezugs- und wieviele Speicherstellen die Unbekannte-Antwort-Tabelle?
Bei 4 Eingängen sind für die Bezugs-Antwort-Tabelle und für die Unbekannte-Antwort-Tabelle 2×2^4 Bytes erforderlich (32). Der Inhalt des BC-Registers ist also 0020.

9. Schritt
Für das IC 74LS54 (Invertierendes AND/OR-Gatter mit 3-2-2-3 Eingängen) gibt die Tabelle 8-2 folgende Daten an: 10 Eingänge, 1 Ausgang und 1 Anschlußpin nicht belegt. Wieviele Speicherplätze benötigen die Bezugs- und die Unbekannte-Antwort-Tabelle?
Die Antwort-Tabellen des Bezugs- und Prüf-ICs benötigen je $2 \times 2^{10} = 2048$ Bytes. Der Platz für beide Tabellen beträgt demnach 4096 Bytes. Die CHPTST-Routine sieht pro Antwort-Tabelle nur 1024 Bytes vor. Um das IC zu prüfen, sind also einige Änderungen erforderlich.

VERSUCH NR. 3

1. Schritt
Dieser Versuch beschäftigt sich nur mit einem Gatter des ICs 74LS08. Wie aus der Anschlußbelegung in Bild 8-1 hervorgeht, sind bei Gatter A die Pins 1 und 2 Eingänge und der Pin 3 ein Ausgang. Da beim Test nur dieses Gatter vom Mikroprozessor berücksichtigt werden soll, lautet das I/O-Steuerwort

1111 1111 1111 1100 oder FFFCH.

Laden Sie dieses Maskenwort (HI-Byte = FF; LO-Byte = FC) in die Speicherstellen MASKW und MASKW + 1. Prüfen Sie, ob die Breakpoints wie im 3. Schritt von Versuch Nr. 1 beschrieben, gesetzt sind.

2. Schritt.
Bringen Sie das IC 74LS08 in Prüfposition und schließen die Versorgungsleitungen an. Da dieses IC nur 14 Anschlußpins hat, müssen PC7 und PD0 mit Masse verbunden sein. Starten Sie das Programm bei CHPTST. Sobald der Nanocomputer® die Breakpointadresse ENDREF anzeigt, prüfen Sie den Inhalt des RC-Registers. Er ist 0008, da der Test nur auf ein Gatter beschränkt ist. Für den Mikrocomputer sind als Eingänge somit auch nur zwei Anschlußpins mit vier verschiedenen Kombinationsmöglichkeiten relevant. Jede Kombinationsmöglichkeit benötigt für die Wahrheitstabelle 2 Bytes, so daß für die Bezugs-Antwort-Tabelle insgesamt 8 Bytes erforderlich sind.

3. Schritt
Prüfen Sie die Bezugs-Antwort-Tabelle ab Adresse 0800 im Lese-/Schreibspeicher. Notieren Sie ab REFIC den Inhalt von 8 aufeinanderfolgenden Speicherstellen. Die gemachten Notizen müssen mit folgender Tabelle übereinstimmen.

Tabelle 8-5. Bezugs-Antwort-Tabelle für ein Gatter des ICs 74LS08.

Speicherstelle	Inhalt	Binäres Äquivalent
REFIC	38	0 0 1 1 1 0 0 0
REFIC + 1	FE	
REFIC + 2	38	0 0 1 1 1 0 0 0
REFIC + 3	FE	
REFIC + 4	38	0 0 1 1 1 0 0 0
REFIC + 5	FE	
REFIC + 6	3C	0 0 1 1 1 1 0 0
REFIC + 7	FE	

Die Bytes an den Speicherstellen REFIC + 1, REFIC + 3, REFIC + 5 und REFIC + 7 haben mit dem Gatter nichts zu tun und werden deshalb auch nicht näher beschrieben.

4. Schritt
Wenn Sie den Code der CHPTST-Routine prüfen, erkennen Sie die Reihenfolge für die Ausgabewerte zu den Pins 1 und 2 des ICs 74LS08:

Prüfwort	Pin 1	Pin 2	Pin 3
0000	0	0	0
0001	1	0	0
0002	0	1	0
0003	1	1	1

Der IC-Ausgang an Pin 3 stellt für den Mikrocomputer einen Eingang dar. Das bei Pin 3 anstehende Bitmuster erscheint in der Tabelle 8-5 bei der dritten Pin-Position an den Speicherstellen REFIC, REFIC + 2, REFIC + 4 und REFIC + 6. Da es sich bei dem Gatter des ICs 74LS08 um ein AND-Gatter handelt, hat dieser Vorgang bestätigt, daß die Bezugs-Antwort-Tabelle stimmt.

Kapitel 9

Z-80 Counter-Timer-Circuit (CTC)

Die CTC (Zähler-Zeitgeber-Schaltung) erleichtert genau wie die PIO das Interfacing zur Z-80-CPU. Die CTC führt Zähl- und Zeittakt-Funktionen aus. Sie kann über vier unabhängige 8-Bit-Kanäle die direkte Verbindung zum Z-80-Daten-BUS herstellen. Abhängig von der Programmierung sind die Kanäle jeweils auf den Zähl- oder Zeittaktbetrieb eingestellt. Beim Zählbetrieb erhält die Schaltung Impulse von einem externen Taktgenerator. Nach einer bestimmten Anzahl gezählter Impulse kann die CTC mit einem Interrupt-Signal die CPU nach Methode 2 unterbrechen. Sowohl der Interrupt-Vektor als auch die Impulszahl bis zum Interrupt sind durch die Programmierung wählbar. Beim Zeittaktbetrieb zählt die CTC in erster Linie die Impulse des Systemtaktgebers Φ. Auch bei dieser Betriebsart kann nach einer vorprogrammierten Impulszahl ein Interrupt-Signal erfolgen. Weil die Zykluszeit des Systemtaktgebers bekannt und in der Regel sehr genau ist, kann die CTC die CPU in genau festgelegten Zeitintervallen unterbrechen. Auf diese Weise mißt die CTC die "Echtzeit", das ist die tatsächlich benötigte Zeit, die man auch mit einer Stoppuhr messen kann. Je nach Programmierung der CTC kann der Taktgeber eine Auflösung zwischen Millisekunden und Sekunden haben.

Folgende Hauptmerkmale sind für das CTC-IC charakteristisch:

- Die CTC ist als N-Kanal-Siliziumgatter in Sperrschichttechnik aufgebaut und in einem 28-Pin DIP-Gehäuse untergebracht. Zur Stromversorgung genügt eine Spannung von +5 V und für die Ansteuerung ist ein 5 V-Taktimpuls ausreichend.
- Alle Ein- und Ausgänge sind TTL-kompatibel.
- Alle vier Kanäle können entweder im Zeittakt-oder Zählbetrieb arbeiten.
- Bei beiden Betriebsarten kann die CPU den Inhalt eines Rückwärtszähler bei Null wieder laden.
- Die Triggerung der Zählvorgänge erfolgt je nach Programmierung entweder mit der positiven oder negativen Impulsflanke.
- Das Auslösen eines Interrupt-Signals bei Null ist softwaregesteuert.
- Zur automatischen Interrupt-Vektorisierung ohne externe Logik ist eine verschachtelte Interrupt-Logik mit Prioritätsmerkmalen vorgesehen.
- Drei Kanäle haben Nullzähl/Zeitsperrausgänge und können außerdem Darlington-Treibertransistoren ansteuern.

Nach Studium des Kapitels können Sie

- alle aufgeführten CTC-Merkmale verstehen und

- die CTC-Merkmale experimentell prüfen.

Das CTC-IC

Zunächst folgt eine kurze Beschreibung der Anschlußbelegung sowie eine Zusammenfassung der CTC-Betriebsarten anhand einer Kurzbeschreibung der Programmierung. Der theoretische Teil ist absichtlich kurz gehalten, damit Sie möglichst schnell mit den praktischen Versuchen beginnen können. Das macht Sie mit der CTC-Anwendung wesentlich vertrauter, als jede theoretische Erläuterung.

D7 . . . D0 – Z-80-Daten-Bus (bidirektional und tristate gepuffert)

Alle Daten- und Befehlswörter zwischen der Z-80-CPU und dem CTC-IC werden über die BUS-Leitungen D7 . . . D0 übertragen. Pro Daten- und Befehlswort ist die Übertragung von insgesamt 8 Bit möglich, wovon D0 das niederwertigste ist.

Die Anschlüsse bilden einen binären 2-Bit-Adresseingang. Mit dem entsprechenden Code kann man einen der vier unabhängigen Kanäle für einen I/O-Schreib- oder Lesevorgang anwählen. Es gilt die Wahrheitstabelle:

	CS1	CS0
Ch0	0	0
Ch1	0	1
Ch2	1	0
Ch3	1	1

$\overline{CE}$ – Chip Enable (Freigabe), Pin 16 (Eingang, low-aktiv)

Im Zustand logisch 0 ist es dem CTC-IC möglich, Steuerwörter, Interrupt-Vektoren oder zeitkonstante Datenwörter während eines I/O-Schreibzyklus vom Daten-BUS anzunehmen oder den Inhalt des Rückwärtszählers bei einem I/O-Lesezyklus zur CPU zu übertragen.

Takt Φ, Systemtaktgeber, Pin 15 (Eingang)

Den Taktgeber benutzt die CTC zur Synchronisation bestimmter Signale.

$\overline{M1}$ – Maschinenzyklus,ein Signal der CPU, Pin 14 (Eingang, low-aktiv)

Bei aktivierten M1- und RD-Signalen ruft die CPU einen Befehl vom Speicher ab. Wenn $\overline{M1}$ und $\overline{IORQ}$ aktiv sind, bestätigt die CPU eine Unterbrechchung. Sie fordert außerdem das CTC-IC auf, einen Interrupt-Vektor auf den Z-80-Daten-BUS zu legen, falls einer der vier Kanäle ein Interrupt mit Prioritätskennung wünscht.

$\overline{IORQ}$ – Eingabe/Ausgabe-Anforderung von der CPU, Pin 10 (Eingang, low-aktiv)

Zusammen mit den Signalen $\overline{CE}$ und $\overline{RD}$ überträgt das $\overline{IORQ}$-Signal Daten und Kanal-Steuerwörter zwischen der CPU und der CTC. Während

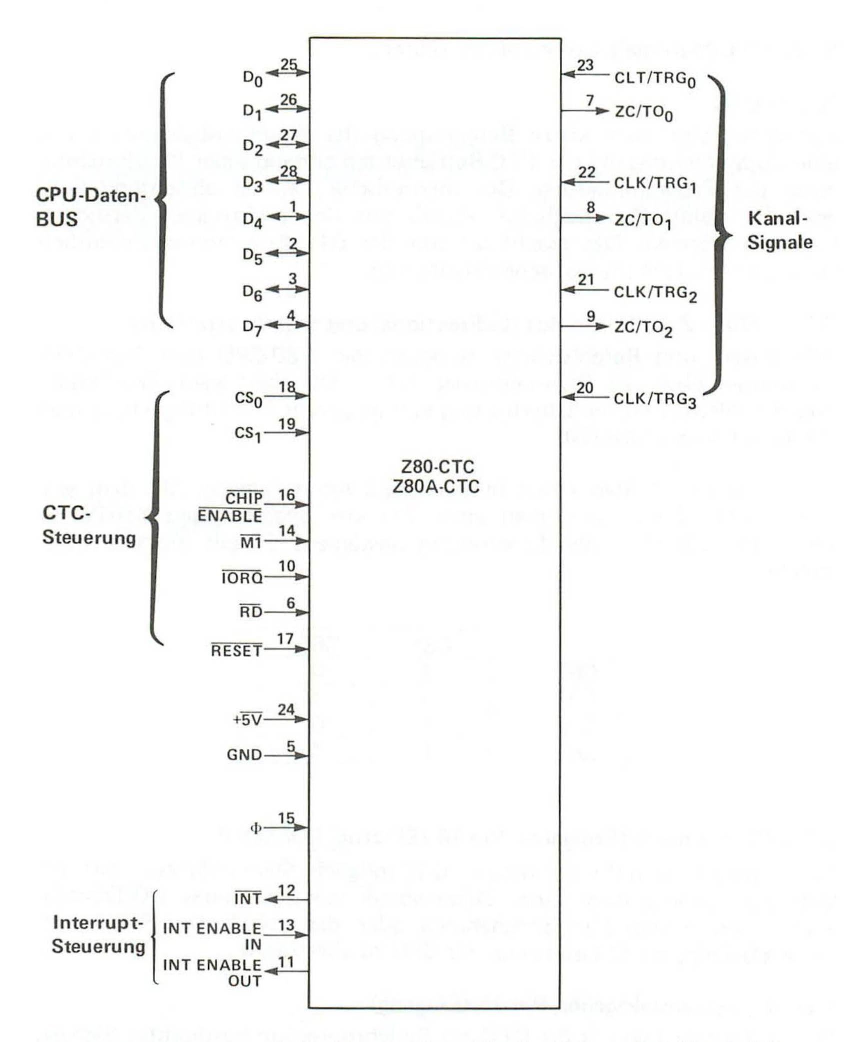


Bild 9-1. Anschlußbelegung des Counter-Timer-Circuits. CS1 . . . CS0 Kanalwähler (Eingang, high-aktiv)

eines CTC-Schreibzyklus müssen $\overline{IORQ}$ und $\overline{CE}$ logisch 0 und $\overline{RD}$ logisch 1 sein. Die CTC empfängt kein spezifisches Schreibsignal, sondern erzeugt stattdessen intern ihr eigenes durch Umkehrung eines gültigen RD-Signals. Bei einem CTC-Lesezyklus müssen $\overline{IORQ}$, $\overline{CE}$ und $\overline{RD}$ aktiv sein, um den Inhalt des Rückwärtszählers auf den Z-80-Daten-BUS legen zu können. Wenn sowohl IORQ als auch $\overline{M1}$ gültig sind, bestätigt die CPU eine Interrupt-Anforderung, und der Interrupt-Kanal mit der höchsten Priorität legt seinen Interrupt-Vektor auf den Z-80-Daten-BUS.

$\overline{\text{RD}}$ – Lesezyklus-Zustand von der CPU, Pin 6 (Eingang low-aktiv)

Das $\overline{\text{RD}}$-Signal wird in Verbindung mit den $\overline{\text{IORQ}}$- und $\overline{\text{CE}}$-Signalen benutzt, um Daten und Kanal-Steuerwörter zwischen der Z-80-CPU und der CTC zu übertragen. Bei einem CTC-Schreibzyklus müssen $\overline{\text{IORQ}}$ und $\overline{\text{CE}}$ wahr und $\overline{\text{RD}}$ falsch sein. Die CTC empfängt kein spezifisches Schreibsignal,sondern erzeugt stattdessen intern ihr eigenes durch Umkehrung eines gültigen $\overline{\text{RD}}$-Signals. Bei einem CTC-Lesezyklus müssen $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ und $\overline{\text{RD}}$ aktiv sein, um den Inhalt des Rückwärtszählers auf den Z-80-Daten-BUS legen zu können.

IEI – Interrupt Enable IN, Pin 13 (Eingang, low-aktiv)

Dieses Signal hilft eine systemweite Interrupt-Verschachtelung bilden. Sie setzt Prioritäten, wenn mehr als ein externes Gerät im System die Fähigkeit zu Unterbrechungen hat. Ist der Pin auf High-Potential, werden keine anderen Interrupt-Geräte höherer Priorität in der Verkettung von der Z-80-CPU bedient.

IEO – Interrupt Enable OUT, Pin 11 (Ausgang, high-aktiv)

Das IEO-Signal wird in Verbindung mit dem IEI-Signal benutzt, um eine systemweite Interrupt-Prioritäts-Verkettung zu bilden. IEO ist nur dann im High-Zustand, wenn dies auch bei IEI der Fall ist und die CPU keine Interrupts von einem der CTC-Kanäle bedient. Somit blockiert dieses Signal ein Gerät mit geringerer Priorität, während ein Interrupt-Gerät mit höherer Priorität von der CPU bedient wird.

$\overline{\text{INT}}$ – Interrupt Request (Unterbrechungsanforderung), Pin 12 (Ausgang, offenes Gatter, low-aktiv)

Dieses Signal wird aktiviert, wenn einer der CTC-Kanäle, der zur Freigabe von Interrupts programmiert ist, eine Nullzählerbedingung in seinem Rückwärtszähler hat.

$\overline{\text{RESET}}$ – Reset (Rücksetzen), Pin 17 (Eingang, low-aktiv)

Dieses Signal stoppt das Zählen aller Kanäle und setzt Kanal-Interrupt-Freigabe-Bits in allen Steuerregistern. Dadurch werden die von der CTC erzeugten Unterbrechungen blockiert. Die ZC/TO und INT-Ausgänge nehmen ihren inaktiven Zustand ein, IEO reflektiert IEI und die Ausgabetreiber des CTC-Daten-BUS nehmen den hochohmigen Zustand ein.

CLK/TRG3 . . . CLK/TRGO – externer Takt/Zeitgeber-Trigger, Pins 20 . . . 23 (Eingang, High- oder low aktiv, vom Benutzer wählbar)

Es gibt vier CLK/TRG-Anschlüsse, die den vier unabhängigen CTC-Kanälen entsprechen. Beim Zählbetrieb vermindert jede Impulsflanke auf diesem Pin den Rückwärtszähler. Beim Zeittaktbetrieb initiiert eine Impulsflanke auf diesem Pin die Zeitgeberfunktion. Der Benutzer kann entweder die ansteigende oder die abfallende Flanke wählen.

ZC/TO2 . . . ZC/TO0-Nullzähler/Zeitsperre, Pins 7 . . . 9 (Ausgang, high-aktiv)

Es gibt drei ZC/TO-Anschlüsse, die den CTC-Kanälen 2 bis 0 entsprechen. (Wegen Gehäuse-Pinbegrenzungen hat Kanal 3 keinen ZC/TO-Pin.) Beim Zähl- oder Zeittaktbetrieb, wenn der Rückwärtszähler in Richtung Null zählt, erscheint an diesem Pin ein high-aktiver Impuls.

CTC - Funktioneller Überblick (Bild 9-2)

Bild 9-2 ist ein funktionelles Blockschaltbild des CTC-ICs. Das Interfacing der Z-80-CPU beinhaltet acht Daten-BUS-Leitungen plus folgender Steuerleitungen: $\overline{\text{M1}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{CS0}}$, $\overline{\text{CS1}}$, und $\overline{\text{CE}}$. Der Daten-BUS trägt alle Daten- und Befehlbytes zwischen der CPU und der CTC. Die Steuerleitungen aktivieren das CTC-IC, wählen einen der vier CTC-Kanäle zur Übertragung ($\overline{\text{CS0}}$ und $\overline{\text{CS1}}$) und bestimmen die Richtung des Datenflusses ($\overline{\text{M1}}$, $\overline{\text{IORQ}}$ und $\overline{\text{RD}}$). Die I/O-Funktion des CPU-BUS kommuniziert mit den anderen Funktionen über den internen CTC-Daten-BUS. Die Interrupt-Steuerfunktion steuert die drei Interrupt-Leitungen: IEI, IEO und $\overline{\text{INT}}$. Die interne Steuerlogik gewährleistet die gesamte Funktion der Synchronisation und Koordination. Die vier Kanalfunktionen stehen jeweils den vier unabhängigen Zähler/Zeitgeberkanälen zur Verfügung, die durch die Zahlen von 0 bis 3 gekennzeichnet sind. Diese vier Kanäle sind als vier abhängige Geräte in der Standard-Z-80-Prioritätsverkettung ausgeführt, wobei die Kanalzahl 0 die höchste Priotität hat. Die Kanäle nehmen eine einmalige Takt/Trigger-Eingabe auf, welche die beim Zählbetrieb zu zählenden Impulse führt. Aufgrund von Gehäusebegrenzungen haben nur die Kanäle 0, 1 und 2 Nullzähler/Zeitsperr-Ausgangsanschlüsse, die nach einer vorgeschriebenen Impulszahl einen high-aktiven Impuls führen.

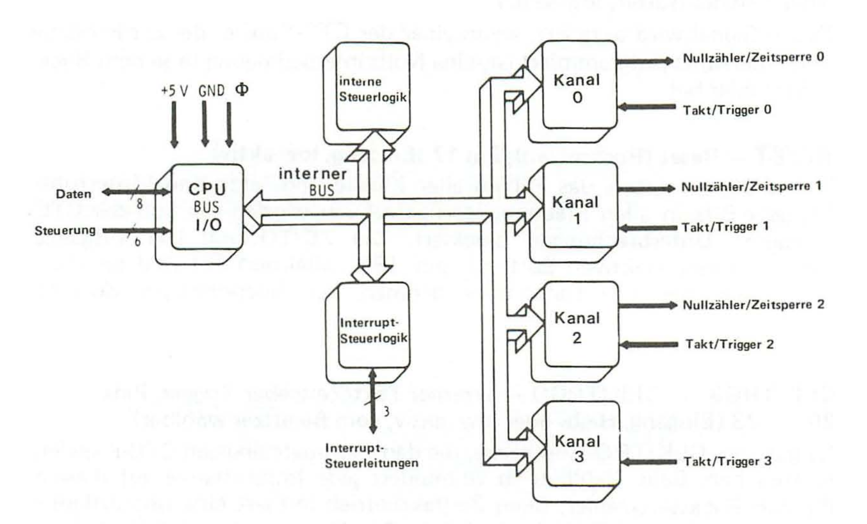


Bild 9-2. Funktionelles Blockschaltbild des CTC-ICs.

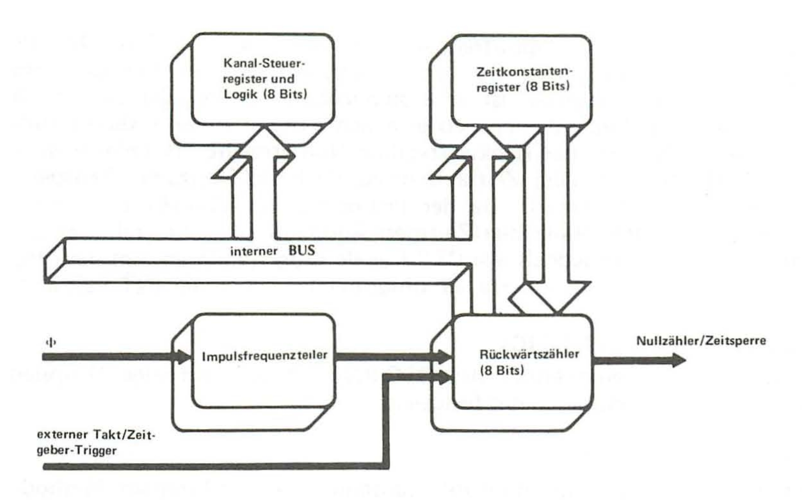


Bild 9-3. Funktionelles Blockschaltbild des CTC-Zähler/Zeitgeberkanals.

Bild 9-3 ist ein funktionelles Blockschaltbild für einen der vier Zähler/ Zeitgeberkanäle. Die Hauptfunktionselemente sind zwei Register, zwei Zähler, Steuerlogik und ein interner BUS. Das Kanal-Steuerregister mit Logikfunktion enthält ein 8-Bit-Register. Die CPU lädt in das Register die Information, welche die Betriebsart des Kanals bestimmt. Der Inhalt dieses Registers bestimmt, ob der Kanal im Zähl- oder Zeittaktbetrieb ist, ob Interrupts freigegeben oder blockiert werden, ob das System im 16er oder 256er Takt (Zeitgeberbetrieb) arbeitet, ob eine positive oder negative Flanke den Zeitgeber triggert (beim Zeittaktbetrieb) oder den Zähler taktet (beim Zählbetrieb) und mehrere andere Gestaltungsparameter, die zu einem späteren Zeitpunkt näher beschrieben sind. Um einen der Zähler/Zeitgaberkanäle zu programmieren, lädt die CPU ihr Kanal-Steuerregister, das von den beiden Kanal-Auswahlanschlüssen CS0 und CS1 (in der Regel an den Leitungen A0 und A1 des CPU-Adreß-BUS) gewählt wird. Unmittelbar nach Laden des Kanal-Steuerregisters lädt die CPU das 8-Bit Zeitkonstantenregister. Der Inhalt dieses Registers bestimmt die Zahl der Impulse, die den Rückwärtszähler auf Null bringen. Die Zahl der Impulse schwankt zwischen 1 bis 256. Wenn der Zähler/ Zeitgeberkanal das erste Mal mit einer Zeitkonstanten geladen wird, übernimmt der Rückwärtszähler automatisch den Registerinhalt. Das gilt auch dann, wenn der Rückwärtszähler Null erreicht hat. Soll das Register eine neue Zeitkonstante laden, während der Kanal zählt oder taktet, wird zuerst der laufende Arbeitsgang beendet. Erst dann lädt der Rückwärtszähler die neue Zeitkonstante. Der Impulsfrequenzteiler teilt den Systemtakt in Impulse mit niedrigeren Frequenzen. Das heißt, der Impulsfrequenzteiler nimmt Taktimpulse von dem Systemtaktgeber an und leitet nur den 16. oder 256. Taktimpuls zum Rückwärtszähler der CTC weiter. Der Rückwärtszähler zählt also entweder mit 1/16 oder 1/256 der System-Taktgebergeschwindigkeit rückwärts. Oder genauer: ist die Taktfolge des Systems t und der Impulsfrequenzteiler auf Teilung durch P pro-

grammiert, leitet der Impulsfrequenzteiler nur jeden t x P-ten Impuls zum Rückwärtszähler. Da der Impulsfrequenzteiler nur Impulse vom Systemtaktgeber annimmt, ist er ausschließlich für den Zeittaktbetrieb von Bedeutung. Beim Zählbetrieb gelangen externe Impulse direkt zum Rückwärtszähler. Hat der Rückwärtszähler Null erreicht, erscheint sowohl beim Zähl- als auch beim Zeittaktbetrieb am Kanal-Nullzähler/Zeitsperr-Anschluß ein Signalimpuls. Da der technologische IC-Aufbau nicht für jeden Kanal einen Nullzähler/Zeitsperr-Anschluß vorsieht, fehlt er bei Kanal 3. Man kann jedoch alle CTC-Kanäle programmieren, um die CPU nach einer Nullzähl/Zeitsperre zu unterbrechen (über die $\overline{INT}$-Leitung).

Programmierung des CTC-ICs

Durch die Programmierung des CTC-ICs sind drei wichtige Gruppen von CTC-Operationsparameter festgelegt:

1. Der Interrupt-Vektor

Die CTC unterstützt die Interrupt-Bearbeitung bei der Interrupt-Methode 2 der Z-80-CPU. Immer, wenn die CTC die CPU unterbricht, erwartet man von ihr einen Gerät-Erkennungscode (siehe Kapitel 6). Dieser Erkennungscode ist das niederwertige Byte einer Vektor-Tabellenadresse, die wiederum auf den Start einer Programmunterbrechung hinweist. Der Erkennungscode wird in das CTC-IC geladen, indem er seine fünf hochwertigen Bits in die I/O-Portadresse einschreibt, die dem CTC-Kanal 0 entspricht. Um der CTC zu signalisieren, das Byte ist zur Bestimmung des Interrupt-Vektors vorgesehen, muß man das Bit D0 auf logisch 0 zurücksetzen. Die Bits D7, D6, D5, D4, D3 und werden alle in das Interrupt-Vektorregister geladen. Den Inhalt der Bits D2 und D1 steuert die CTC-Interrupt-Steuerlogik Wenn ein besonderer Interrupt-Kanal den Interrupt-Vektor auf den Z-80-CPU-Daten-BUS legen muß, liefert die Interrupt-Steuerlogik der CTC automatisch einen binären Code in den Bits D1 und D2. Daraus kann man erkennen, welcher der vier CTC-Kanäle zu bedienen ist. Bild 9-4 zeigt das Verhältnis zwischen dem Inhalt D1 und D2, den CTC-Kanälen und den resultierenden Interrupt-Vektoren.

2. Das Kanal-Steuerwort

Das Kanal-Steuerwort wird von der CPU in das Kanal-Steuerregister geladen, indem sie eine normale I/O-Schreibfolge zur Portadresse der entsprechenden CTC ausführt. Die spezielle CTC wählt ein aktives $\overline{CE}$-Signal aus.

Innerhalb der gewählten CTC legt die 2-Bit-Adresse auf den Leitungen CS0 und CS1 einen individuellen Kanal fest. Falls das Wort auf dem Daten-BUS der CPU Bit D0 auf logisch 1 gesetzt hat, erkennt der CTC-Kanal das Byte als ein Kanal-Steuerwort und lädt es in das Kanal-Steuerregister. Die Bits D1 bis D7 haben folgende Bedeutung:

D7: Interrupt-Freigabe – Wenn D7 gesetzt ist, wird der CTC-Kanal freigegeben, um jedesmal eine Unterbrechung zu erzeugen, wenn der Rückwärtszähler Null erreicht. Ist D7 auf logisch 0 zurückgesetzt, wird bei einer Nullzähl/Zeitsperre keine Unterbrechung erzeugt. Bei Freigabe der Unterbrechungen muß ein Interrupt-Vektor vor Beginn der Operation in

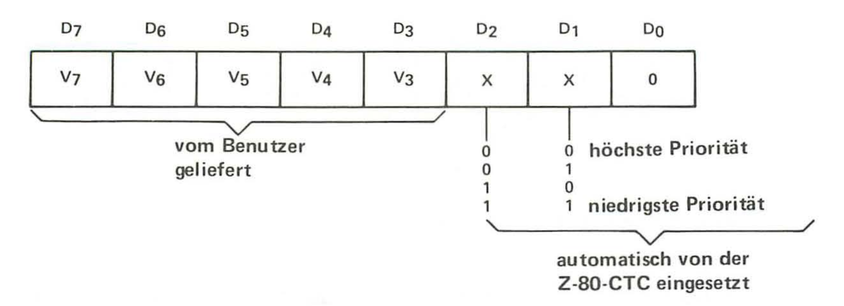


Bild 9-4. Kodierung der Bits D1 und D2 des CTC-Vektorregisters.

das Interrupt-Vektorregister geladen sein.

D6: Modus-Auswahl: Ist D6 logisch 1, arbeitet die CTC im Zählmodus, andernfalls im Zeittaktmodus.

D5: Bereich des Impulsfrequenzteilers – Das Bit D5 ist nur für die Zeittaktmethode zum Aktivieren des Impulsfrequenzteiles auf Teilung des Systemtaktes durch 16 oder 256 von Bedeutung. Logisch 1 entspricht 256 und logisch 0 = 16.

D4: Flankenwahl – Dieses Bit bestimmt, welche Flanke des Taktimpulses die Zeittaktoperation startet oder den Inhalt des Rückwärtszählers bei der Zählmethode vermindert (dekrementiert). Bei logisch 1 wird die positive und bei logisch 0 die negative Flanke benutzt.

D3: Trigger: Dieses Bit ist nur für den Zeittaktbetrieb relevant. Es unterscheidet zwei Triggermöglichkeiten für den Operationsbeginn des Zeitgebers. Logisch 1 bestimmt den Trigger für das externe Gerät über die externe Takt/Zeitgeber-Triggerleitung. Bei logisch 0 startet der Zeitgeber die Operation mit der ansteigenden Flanke von T2 des Maschinenzyklus. Der Impuls lädt die Zeitkonstante. Erfolgt die Triggerung extern, wird der Impulsfrequenzteiler zwei Taktzyklen später vermindert, falls zur Aktivierung des Takt/Zeitgeber-Triggersignals (CLK/TRG) die Setzzeit 130 ns beträgt. Das heißt, CLK/TRG muß mindestens 130 ns lang logisch 1 sein, bevor die nächste ansteigende Flanke von Φ für die Verzögerung zwischen dem externen Trigger und der Aktivierung des Impulsfrequenzteilers zwei Taktzyklen dauert.

D2: Lade-Zeitkonstante: Das aktivierte Bit D2 setzt den CTC-Kanal, um auf eine Zeitkonstante zu warten, die auf dem Kanal als nächste Byteausgabe folgt.

D1: Rücksetz-Kanal (Reset): Wird Bit D1 gesetzt, hört der Kanal auf zu zählen bzw. zu takten. Die Bits im Kanal-Steuerregister bleiben unverändert. Ist auch Bit 2 gesetzt, nimmt der Kanal die Operation wieder auf, wie es das Kanal-Steuerwort bestimmt, nachdem die Zeitkonstante (nächstes Byte) geladen worden ist.

3. *Die Zeitkonstante*

Vor Operationsbeginn muß man einen CTC-Kanal mit einer Zeitkonstanten laden. Die Zeitkonstante kann jeden Wert zwischen 1 und 256 annehmen,

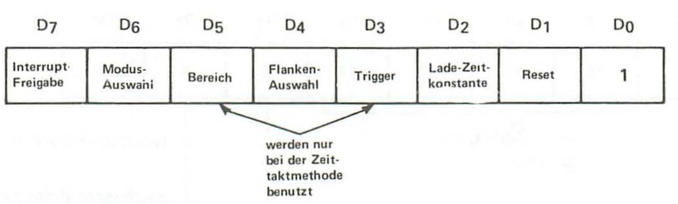


Bild 9-5. Format des Kanal-Steuerregisters.

wobei das Datenbyte 00H dem Wert 256 entspricht. Ein CTC-Kanal lädt die Zeitkonstante in zwei Stufen.

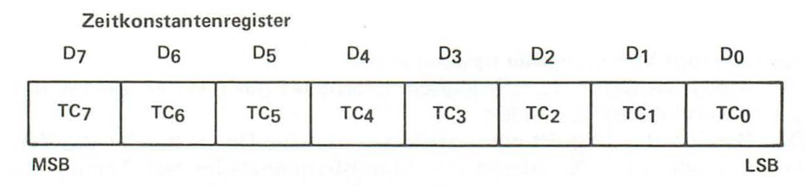


Bild 9-6. Format des Zeitkonstantenregisters.

Zunächst geht ein Kanal-Steuerwort zum Kanal und das Lade-Zeitkonstantenbit D2 auf logisch 1. Dann lädt der Kanal die nächste Byteausgabe, wie in dem bestimmten Kanal-Zeitkonstantenregister.

CTC-IC und Z-80-Interrupt-Steuerlogik

Wie bereits erwähnt, kann man die CTC so programmieren, daß sie eine Interrupt-Bearbeitung von der CPU anfordert. Allerdings nur unter den Voraussetzungen, daß die CTC-Interrupts freigegeben und ein Interrupt-Vektor in das CTC-Interrupt-Vektorregister geladen ist. In dem Kapitel über die PIO-ICs haben Sie bereits die Bedeutung einer *Verkettung* kennengelernt. Sie ist so realisiert, das man die Pins mehrerer PIO-ICs untereinander verbindet. Die CTC (wie auch alle anderen ICs in der externen Z-80-Familie) ist ebenfalls mit IEI- und IEO-Pins ausgerüstet. Somit können die Kanäle im CTC-IC vier aufeinanderfolgende Speicherstellen in einer Z-80-Interrupt-Verkettung besetzen. Der folgende Abschnitt geht auf die Z-80-Interrupt-Verkettung näher ein. Dabei wird auf die CTC-ICs in den meisten Fällen als Elemente der Verkettungsbeispiele Bezug genommen. Die meisten Prinzipien sind jedoch auch auf alle ICs der Z-80-Familie anwendbar. Die Schaltung in Bild 9-7 zeigt, wie Z-80-Interrupt-Steuerlogik in allen der zur Z-80-Familie gehörenden ICs realisiert wird. Die drei kritischsten Signale sind IEI, IEO und INT. Nachstehend eine Zusammenfassung mehrerer wichtiger Fakten über diese Schaltung.

– Eine Notwendigkeit zur Bearbeitung, repräsentiert durch das NEED SERVICE-Signal, wird aufgrund der speziellen Eigenschaften des externen Z-80-ICs bestimmt. Für die CTC steht ein NEED-SERVICE-Impuls zur Verfügung, wenn Interrupts freigegeben sind und ein Kanal-Rückwärtszähler Null erreicht.

- Ein aktiviertes NEED-SERVICE-Signal setzt das Interrupt-Vorbereitungs-Flipflop (sein Q-Ausgang geht auf logisch 1). Die Logikpegel anderer Signale bleiben hierauf ohne Einfluß.
- Falls IEI sich im High-Zustand befindet, setzt ein Interrupt-Bestätigungszyklus ($\overline{\text{M1}}$ und $\overline{\text{IORQ}}$) das Interrupt-Vorbereitungs-Flipflop zurück; der genannte Zyklus, setzt das Interrupt-Bearbeitungs-Flipflop. Weil IEI im High-Zustand sein muß, wird nur das Gerät vom Interrupt-Bestätigungszyklus beeinflußt, dessen Unterbrechung zu bedienen ist.
- Die Interrupt-Vektoradresse für das zu bedienende Gerät wird während eines Interrupts-Bestätigungszyklus auf den Daten-BUS gesetzt. Damit nur der Interrupt-Vektor des entsprechenden Gerätes zum Daten-BUS gelangt, sind die Geräte-BUS-Puffer nur unter folgenden Bedingungen freigegeben:

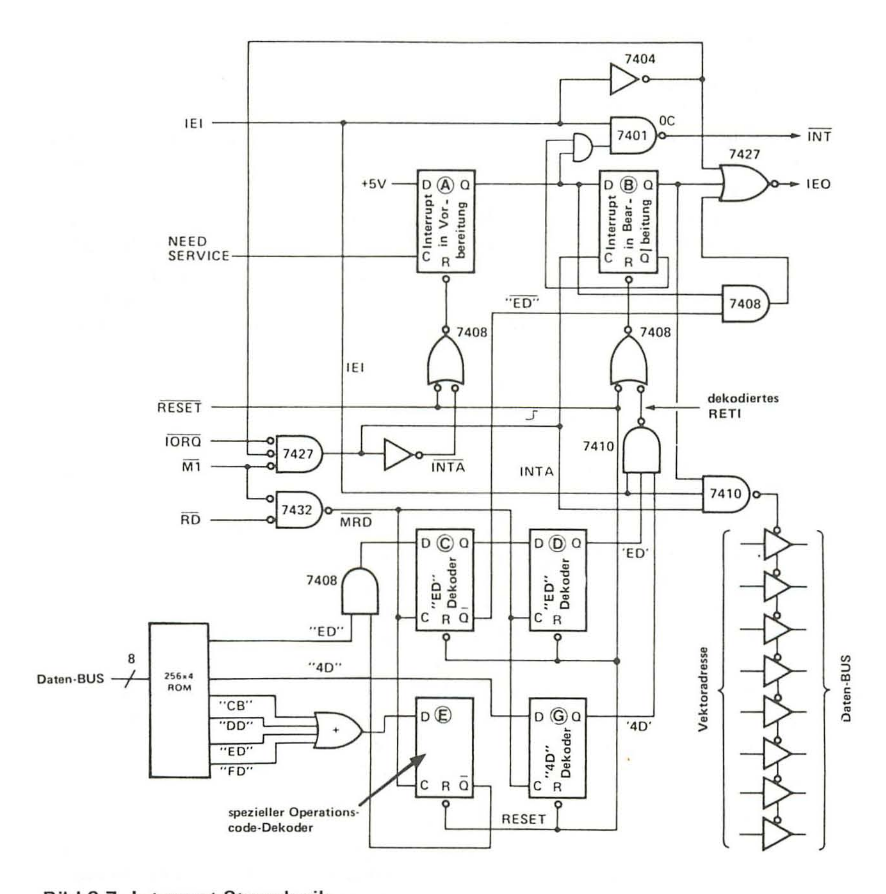


Bild 9-7. Interrupt-Steuerlogik.

1. Das Interrupt-Bearbeitungs-Flipflop des entsprechenden Gerätes wird gesetzt.
2. Die IEI-Eingabe des Gerätes ist im High-Zustand.
3. Interrupt-Bestätigungssignal ist aktiv.

Falls IEI im Low-Zustand ist, aktiviert das Gerät nicht das $\overline{INT}$-Signal. Das Gerät kann $\overline{INT}$ nur unter folgenden Voraussetzungen aktivieren:
1. IEI muß im High-Zustand sein.
2. Das Interrupt-Vorbereitungs-Flipflop muß gesetzt sein.
3. Das Interrupt-Bearbeitungs-Flipflop muß zurückgesetzt sein.

Eine bereits bei den PIO-Versuchen experimentell beobachtete Tatsache: ein Gerät (ein Kanal) muß sich nicht selbst unterbrechen. Auch muß einem Gerät eine Interrupt-Anforderung zugeteilt sein, bevor ein Interrupt erfolgt – wenn auch die Zuteilung nur kurzfristig ist.

Ein High-Zustand von IEO setzt folgendes voraus:
1. IEI muß im Zustand logisch 1 sein.
2. Das Interrupt-Bearbeitungs-Flipflop muß zurückgesetzt sein.
3. Das Interrupt-Vorbereitungs-Flipflop muß zurückgesetzt sein.

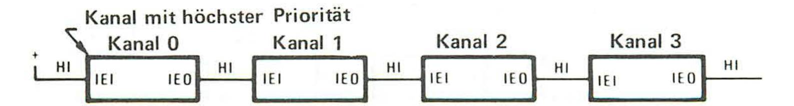
ODER

Der Operationscode ED muß während eines $\overline{M1}$-Zyklus auf dem Daten-BUS zur Verfügung stehen.

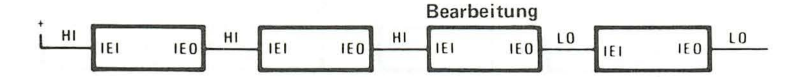
Die Datenleitungen werden auf das Vorhandensein eines RETI-Befehls (ED4D) kontrolliert. Um die vorstehenden Fakten zu untersuchen, nachstehend zwei Beispiele:

Beispiel 1: Keine Z-80-Interrupt-Freigabe. Betrachten Sie die Vier-Geräte-Verkettung in Bild 9-8A. Alle IEI- und IEO Signale sind im High-Zustand, es stehen also keine Unterbrechungen bevor. Der IEI-Eingang des Kanals mit der höchsten Priorität (Kanal 0) ist an +5 V angeschlossen. Ein Interrupt auf diesem Kanal unterbricht und verschiebt die Interrupt-Bearbeitung der Kanäle 1, 2 oder 3, falls Z-80-Interrupts freigegeben sind.

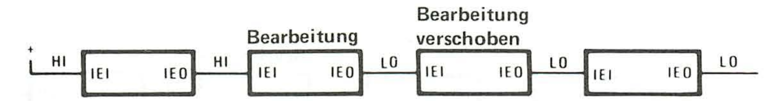
Betrachten Sie folgende Ereignisse:
1. Kanal 2 fordert eine Unterbrechung an und erhält die Bestätigung. Bild 9-8B zeigt das Ergebnis der Situation. Der IEI-Eingang von Kanal 2 ist high (HI), während der IEO-Ausgang low (LO) ist, wodurch auch der IEI-Eingang von Kanal 3 auf low geht. Kanal 3 gibt diese Änderung am Eingang direkt zum Ausgang weiter. Bei evtl. nachfolgenden Geräten in der Verkettung "unterhalb" Kanal 3 werden die IEI- und IEO-Leitungen in gleicher Weise in den Low-Zustand versetzt. Es ist also nur bei Kanal 2 der Eingang HI und der Ausgang LO. Alle folgenden Kanäle führen am Ein- und Ausgang LO-Signal; alle vorhergehenden Kanäle führen am Ein- und Ausgang HI-Signal.
2. Kanal 1 fordert eine Interrupt-Bearbeitung an. Da Kanal 1 eine höhere Priorität in der Verkettung einnimmt unterbricht Kanal 2 sofort die Bearbeitung. Die Interrupt-Anforderung von Kanal 1 wird bestätigt und mit der Bearbeitung begonnen. Das Ergebnis der Situation ist in Bild 9-8C dargestellt.
3. Kanal 1 beendet die Bearbeitung. Alle Kanäle in der Verkettung kontrollieren den Daten-BUS auf einen RETI-Befehl. Ist der Befehl vorhanden, revidiert jeder Kanal den Zustand seiner entsprechenden IEO-Leitungen, wie in Bild 9-8D gezeigt. Da kein weiteres Interrupt-



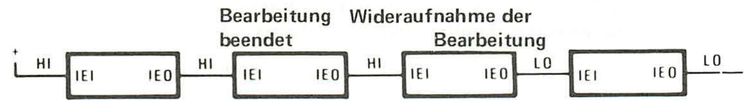
(A) Prioritätsverkettung vor einer Interrupt-Bearbeitung.



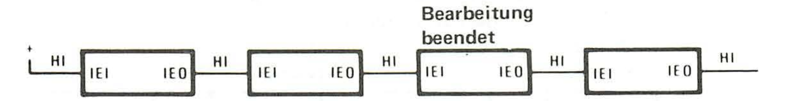
(B) Interrupt-Anforderung durch Kanal 2 und Bestätigung.



(C) Kanal 1 unterbricht und verschiebt die Bearbeitung von Kanal 2.



(D) Bearbeitung von Kanal 1 mit RETI-Befehl beendet, Rückgabe an Kanal 2.



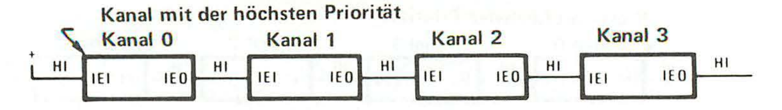
(E) Kanal 2 führt die Interrupt-Bearbeitung weiter und beendet sie mit dem RETI-Befehl.

Bild 9-8. Verkettung – Beispiel Nr. 1.

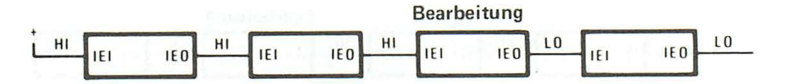
Signal mit höherer Priorität anliegt, setzt Kanal 2 die unterbrochene Bearbeitung fort.

4. Die Interrupt-Bearbeitung von Kanal 2 endet mit dem RETI-Befehl. Wieder tasten alle Kanäle den RETI-Befehl auf dem Daten-BUS ab und revidieren ihre IEO-Leitungen entsprechend. Das Ergebnis ist in Bild 9-8E dargestellt.

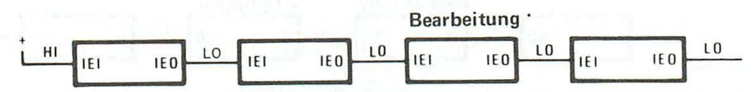
Beispiel 2: Nicht wieder freigegebene Interrupts nach der ersten Unterbrechung. Betrachten Sie die vier Kanäle in Bild 9-9A. Wieder hat Kanal 0 die höchste Priorität. Es sind keine Unterbrechungen in Bearbeitung noch stehen welche aus. Dieses Beispiel folgt der Logikschaltung in Bild 9-7. Insbesondere gilt die Aufmerksamkeit den Zuständen des Interrupt-Vorbereitungs-Flipflops und des Interrupt-Bearbeitungs-Flipflops.
Betrachten Sie folgende Ereignisfolge:



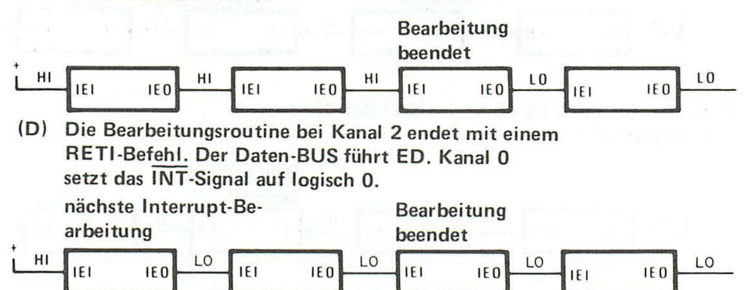
(A) Prioritätsverkettung vor einer Interrupt-Bearbeitung.



(B) Interrupt-Anforderung durch Kanal 2 und Bestätigung.



(C) Interrupt-Anforderung durch Kanal 0; sie wird von der CPU ignoriert. Kanal 0 setzt das Interrupt-Vorbereitungs-Flipflop und den Ausgang auf low.



(D) Die Bearbeitungsroutine bei Kanal 2 endet mit einem RETI-Befehl. Der Daten-BUS führt ED. Kanal 0 setzt das $\overline{INT}$-Signal auf logisch 0.

(E) Auf dem Daten-BUS steht anstelle von ED der Befehl 4D. Kanal 0 ist für die nächste Interrupt-Bearbeitung vorbereitet und führt sie durch, sobald die maskierbaren Interrupts wieder freigegeben sind.

Bild 9-9. Verkettung – Beispiel Nr. 2.

1. Kanal 2 empfängt eine Interrupt-Anforderung. Dies löst folgende Ereignisse aus:

a) Das Interrupt-Vorbereitungs-Flipflop erhält einen Taktimpuls; das Flipflop ist gesetzt.

b) Da der IEI-Anschluß von Kanal 2 HI-Signal führt, wird das Interrupt-Bearbeitungs-Flipflop zurückgesetzt und das Interrupt-Vorbereitungs-Flipflop gesetzt sowie $\overline{INT}$ aktiviert.

c) Der IEO-Anschluß von Kanal 2 geht in den Low-Zustand. Dieser Low-Zustand überträgt sich auf alle folgenden Glieder in der Verkettung (siehe Bild 9-9B).

Somit ist eine Interrupt-Anforderung erfolgt.

2. Die Z-80-CPU tastet die $\overline{INT}$-Leitung im Low-Zustand, prüft das eigene Interrupt-Flipflop IFF1 (das zurückgesetzt sein soll) und führt einen Interrupt-Betsätigungszyklus aus. Durch Initiierung der Interrupt-Bearbeitung von Kanal 2 setzt die CPU das Flipflop IFF1 zurück. Dadurch sind maskierbare Unterbrechungen blockiert. Für das Beispiel gilt die Annahme, daß die Interrupt-Service-Routine des Kanals 2 *nicht* wieder freigegeben wird.
3. Das aktive $\overline{INTA}$-Signal setzt das Interrupt-Vorbereitungs-Flipflop des Kanals 2 zurück, während das Interrupt-Bearbeitungs-Flipflop gesetzt wird. Es ändern sich weder die Signale der IEI- noch die IEO-Leitungen.
4. Kanal 0 erhält ein NEED-SERVICE-Signal, welches eine Interrupt-Bearbeitung anfordert. IEO geht in den Low-Zustand und das Interrupt-Vorbereitungs-Flipflop wird gesetzt. Das LO-IEO-Signal von Kanal 0 überträgt sich auf die ganze Verkettung (Bild 9-9C).
5. Die Folge auf Schritt 4 ist eine Interrupt-Anforderung. Die CPU ignoriert diese Anforderung, weil maskierbare Unterbrechungen blockiert sind. Somit bleibt Kanal 2 IN BEARBEITUNG, obwohl Kanal 0 ein Gerät mit höherer Priorität ist.
6. Die Bearbeitung von Kanal 2 endet mit einem RETI-Befehl. Maskierbare Interrupts sind noch immer nicht freigegeben.
7. ED erscheint während eines $\overline{M1}$-Zyklus auf dem Daten-BUS. Deshalb wechseln die IEO-Ausgänge für die Kanäle 0 und 1 in den High-Zustand. Die IEO-Ausgänge der Kanäle 2 und 3 bleiben hingegen low, weil das Interrupt-Bearbeitungs-Flipflop von Kanal 2 noch gesetzt ist. Den Zustand zeigt Bild 9-9D. Sobald IEI für die Geräte mit gesetztem Interrupt-Vorbereitungs-Flipflop und rückgesetztem Interrupt-Bearbeitungs-Flipflop im HI-Zustand ist, versetzen diese Geräte die $\overline{INT}$-Leitung solange in dem LOW-Zustand, wie ED auf dem Daten-BUS bleibt. Das einzige Gerät in diesem Beispiel mit dieser Situation ist Kanal 0.
8. Das Byte 4D erscheint während des nächsten $\overline{M1}$-Zyklus auf dem Daten-BUS (ein RETI-Befehl hat den 2-Byte-Operationscode ED4D). Damit ein Gerät den Befehl 4D tastet und das Interrupt-Bearbeitungs-Flipflop zurücksetzt, muß der IEI-Eingang des Gerätes in der Verkettung im HI-Zustand sein. In Bild 9-9D trifft dies auf Kanal 2 zu. Nur da ist der IEI-Pin im HI-Zustand und das Interrupt-Bearbeitungs-Flipflop gesetzt. Das Flipflop wird also zurückgesetzt.
9. Sobald der ED-Befehl nicht mehr auf dem Daten-BUS ansteht, versetzen alle Geräte mit nichtbearbeiteten Interrupts den IEO-Ausgang in den Low-Zustand. So auch Kanal 0, was sich wieder auf die gesamte Verkettung überträgt.
10. Die Situation in Bild 9-9E bleibt solange bestehen, bis maskierbare Interrupts wieder freigegeben sind.
11. Ausnahme: Die Interrupts werden unmittelbar vor Ausführung des RETI-Befehls in der Programmunterbrechung freigegeben. In diesem Fall versetzen alle Geräte mit gesetztem Interrupt-Vorbereitungs-Flipflops ihre entsprechenden IEO-Leitungen sofort in den Low-Zustand, nachdem ED den Daten-BUS verlassen hat und 4D auf

dem Daten-BUS ist. Deshalb darf nur das Gerät den Interrupt-Vektor während des folgenden Interrupt-Bestätigungszyklus auf den Daten-BUS legen, dessen IEI-Eingang high ist. Das ist nur das Gerät mit der höchsten Priorität.

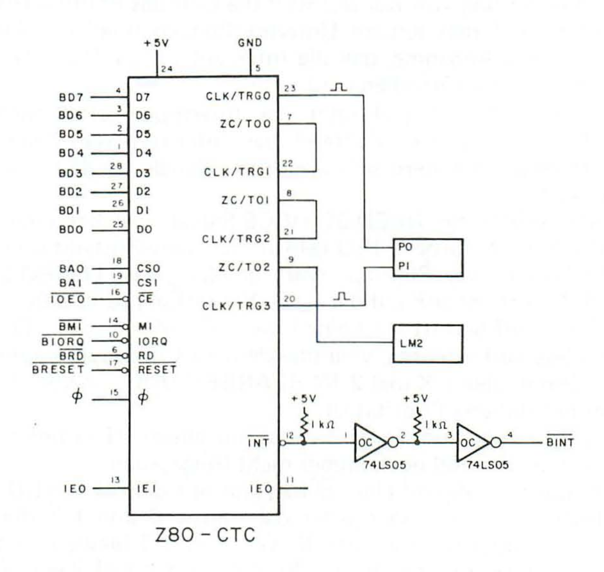


Bild 9-10. Versuchsaufbau.

Einführung in die Versuche

Bei den Versuchen in Kapitel 9 verbinden Sie ein CTC-IC über die Experimentierplatine mit dem Nanocomputer®.
Nachstehend eine Zusammenfassung der CTC-Versuche:

Versuch Nr.	Bemerkung
1	Demonstriert den Zählbetrieb der CTC.
2	Liest den Inhalt des Rückwärtszählers während des CTC-Betriebs.
3	Zeigt den Betrieb der CTC nach der Zeittaktmethode. Die CTC arbeitet als Stoppuhr.

VERSUCH NR. 1

In diesem Versuch arbeitet Kanal 1 des CTC-ICs im Zählerbetrieb.

Programme: MAIN, SERV1 und INITC1

Neben den bereits bekannten Routinen MAIN und SERV1 wird in diesem Versuch das folgende Programm benutzt:

Objekt-Code		Quell-Code	Bemerkung
		NAME INITC1	
ED5E	INITC1:	IM2	; Interrupt-Modus 2
21000F		LD HL,TABLE	; Adresse der Vektortabelle
7C		LD A,H	; HI-Adreßbyte
ED47		LD I,A	; Interrupt-Register setzen
FD216E02		LD IY,SERV1	; Adresse der Service-Routine
FD221A0F		LD (TABLE+1AH),IY	; Tabelle einsetzen
3E18		LD A,18H	; Interrupt-Vektor
D310		OUT (10H),A	; in CTC-Kanal 0 laden
08		EX AF,AF'	; Format für CONVDI setzen
3E40		LD A,40H	
08		EX AF,AF'	
3EC7		LD A,0C7H	; Kanal-Steuerwort
D311		OUT (11H),A	
3E05		LD A,05H	; Zeitkonstantenregister setzen
D311		OUT (11H),A	
C3C302		JP MAIN	; zur MAIN-Routine zurück

1. Schritt

Bauen Sie die für diesen Versuch vorgesehene Schaltung auf (Bild 9-10). Die Adreß-Dekodierung für diese Schaltung ist ähnlich wie beim PIO-IC. Die Zwei Leitungen CS0 und CS1 dekodieren die Adreßleitungen BA0 und BA1. Die $\overline{\text{CE}}$-Eingabe ist am Signal $\overline{\text{IOE0}}$ angeschlossen. Um das Signal IEO0 zu aktivieren, müssen BA2 bis BA7 die folgenden Werte einnehmen:

BA7	BA6	BA5	BA4	BA3	BA2
0	0	0	1	0	0

Die beiden Adreßleitungen BA1 und BA0 dienen als Kanal-Auswahleingänge. Für die Adressierung der vier CTC-Kanäle gilt:

Kanal	Hex-Adresse
0	10
1	11
2	12
3	13

Die Daten-BUS-Leitungen der CPU sind direkt mit den CTC-Daten-BUS-Leitungen verbunden. Die CTC-Steuerleitungen $\overline{\text{M1}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ sind direkt an den gepufferten Z-80-Anschlußpins mit derselben Bezeichnung angeschlossen. Der gepufferte Z-80-Taktgeber BΦ ist mit dem Taktgebereingang an Pin 15 der CTC verbunden. Die CTC-$\overline{\text{INT}}$-Leitung ist mit $\overline{\text{BINT}}$ verbunden. IEO-Pin von PIO2 auf der PC-Platine des Nanocomputers® ist an den IEI-Eingang des CTC-ICs angeschlossen. Somit ist die Priorität der vier CTC-Kanäle niedriger als die von PIO2. Der IEO-Anschluß am CTC bleibt frei. Die CLK/TRG1-Leitung (für Kanal 1) ist an Impuls-

geber P0 und die ZC/TO1-Leitung an eine LED angeschlossen. Schalten Sie die Stromversorgung erst dann ein, wenn beim CTC-IC alle Verbindungen einschließlich der Stromversorgung hergestellt sind.

2. Schritt

Die in diesem Versuch benutzte Software besteht aus drei Routinen: INITC1, MAIN und SERV1. MAIN und SERV1 sind bekannt. MAIN ist das Hauptprogramm und SERV1 eine Interrupt-Service-Routine. Sie übernimmt die Steuerung, wenn das CTC-IC ein Interrupt-Signal erzeugt. INITC1 ist eine Initialisierungs-Routine für diesen Versuch mit folgenden Funktionen:

1. Setzen der Interrupt-Methode Modus 2, vektorisierte Interrupts.
2. Initialisierung der Z-80-Interrupt-Register zum hochwertigen Byte der Tabelle.
3. Laden der Adresse der Interrupt-Service-Routine SERV1 in die richtige Speicherstelle der Interrupt-Vektortabelle.
4. Laden des Interrupt-Vektors zur Portadresse des CTC-Kanals 0. Der Interrupt-Vektor wird immer in den Kanal 0 geladen. Die fünf höherwertigen Bits sind von der Software-Steuerung abhängig, während die drei niederwertigen Bits automatisch durch die CTC bestimmt sind.
5. Setzen des A-Registers für die Subroutine CONVDI auf 40H.
6. Schreiben des Kanal-Steuerwortes zum CTC-Kanal 1.
7. Schreiben des Zeitkonstanten-Datenwortes zum CTC-Kanal 1.
8. Springen zur MAIN-Routine.

Bild 9-11 zeigt den Steuerungsablauf zwischen INITC1, MAIN und SERV1. INITC1 programmiert den CTC-Kanal 1 für den Betrieb nach der Zählmethode mit freigegebenen Interrupts, indem sie den Inhalt der drei Register innerhalb des CTC-ICs bestimmt:

1. Das CTC Interrupt-Vektorregister
2. Das Steuerregister des Kanals 1
3. Das Zeitkonstantenregister des Kanals 1

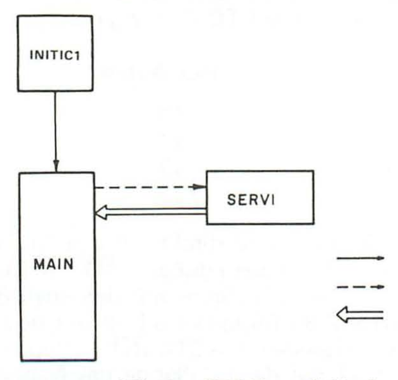


Bild 9-11. Steuerungsablauf für den CTC-Versuch Nr. 1
1 = Beendigung der INITC1-Routine
2 = Antwort auf Interrupt
3 = Rücksprung von Interrupt

Interrupt-Vektorregister

Es ist wünschenswert, einen Interrupt-Vektor auszuwählen, der mit der bestehenden Interrupt-Vektortabelle aus früheren Versuchen übereinstimmt. Zu diesem Zwecke muß die auszuwählende Adresse des Interrupt-Vektors größer sein als 0F11, da sich die Adresse von SERV1F bei 0F11 befindet.
Da die CTC automatisch festlegt, daß die drei niederwertigen Bits für den Betrieb des Kanals 0 logisch 0 sein müssen, ist der nächstverfügbare Tabelleneintrag 0F18H. Somit schreibt INITC1 18 zur Gatter-Adresse des CTC-Kanals 0, um das niedrigwertige Byte des Interrupt-Vektors ins CTC-Interrupt-Vektorregister zu laden (INITC1 initialisiert Register I der Z-80-CPU zu 0F).

Kanalsteuerregister

INITC1 gibt C7H an Port 11 von CTC-Kanal 1 aus. Die CTC erkennt dieses Byte als ein Kanalsteuerwort, weil das Bit D0 gesetzt ist. Die Bedeutung der einzelnen Bits in diesem Wort ist folgende:

Bit D7 = "1": CTC-Interrupts sind freigegeben. Daher aktiviert die CTC die $\overline{\text{BINT}}$-Leitung, wenn der Rückwärtszähler des Kanals 1 Null erreicht.
Bit D6 = "1": Kanal 1 arbeitet nach der Zählermethode.
Bit D5 wird beim Zählbetrieb nicht benutzt.
Bit D4 = "0": Die negativen Flanken der Impulse auf der CLK/TRG1-Leitung vermindern den Inhalt des Rückwärtszählers.
Bit D3 wird im Zählbetrieb nicht benutzt.
Bit D2 = "1" Es folgt das Zeitkonstanten-Byte.
Bit D1 = "1" Die CTC kann mit der Operation beginnen, nachdem die Zeitkonstante geladen ist.

Zeitkonstantenregister

Die nächste Byteausgabe an Port 11 ist die Zeitkonstante. Der von INITC1 gesetzte Wert der Zeitkonstante ist 05. Die Zeitfolge zum Schreiben der jeweiligen Bytes aus dem vorigen Programm in das CTC-IC ist aus Bild 9-12 ersichtlich.

3. Schritt

Mit der Ausführung von INITC1 beginnen. Impulsgeber P0 viermal auslösen. Was passiert?
Die Bearbeitungsanzeige ändert sich nicht. Die externe Impulsquelle P0 hat den Rückwärtszähler des CTC-Kanals 1 veranläßt, viermal zu dekrementieren.

4. Schritt

Lösen Sie mit dem Impulsgeber weitere 5 Impulse aus.
Sobald die CTC die negative Flanke des 5. Impulses wahrgenommen hat, folgt eine Unterbrechung. Die Interrupt-Service-Routine SERV1 erhöht den Bearbeitungszähler 10x und kehrt dann zur MAIN-Routine

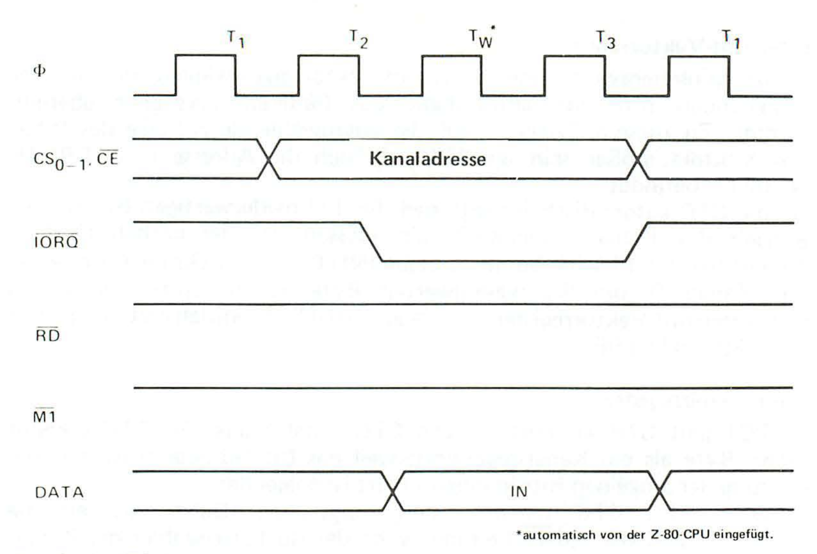


Bild 9-12. CTC-Schreibzyklus.

zurück. Die an ZC/TO1 angeschlossene LED ändert ihren Zustand nicht, da die Aktivierung von TO1 zu kurz ist, um die Veränderung festzustellen. Um den Impuls an ZC/TO1 zu beobachten, können Sie mit einem nachgeschalteten Flipflop das aktivierte Signal auffangen.

5. Schritt

Impulsgeber P0 fünfmal auslösen. Während die Routine SERV1 abläuft, P0 schnell nochmals fünfmal auslösen. Die erste von der CTC erzeugte Unterbrechung wird bearbeitet, aber die zweite nicht. Somit speichert die CTC keine zweite Unterbrechung, die auf einem Kanal erzeugt wird, der bereits ein Interrupt-Signal verarbeitet.

6. Schritt

Impulsgeber P0 fünfmal auslösen. Während SERV1 aktiviert ist, P0 schnell nochmals fünfmal auslösen. Sobald SERV1 an die MAIN-Routine übergeben hat, P0 noch einmal betätigen. – Es wird eine Unterbrechung erzeugt. Offensichtlich funktioniert der Rückwärtszähler des CTC-Kanals 1, während sich noch eine Unterbrechung in Bearbeitung befindet, obwohl die CTC eine unbearbeitete Unterbrechung nicht auffängt.

Lassen Sie Schaltung für den nächsten Versuch bestehen!

VERSUCH NR. 2

Der Versuch zeigt eine Schaltung und die geeignete Softwareroutine, um den Inhalt des Rückwärtszählers vom CTC-Kanal 1 mit Hilfe eines Interrupts von Kanal 0 zu lesen und anzuzeigen.

ANMERKUNG: Die Schaltung in Bild 9-13 stellt *Ergänzungen* zur Schaltung aus Bild 9-10 dar.

Programme: MAIN, SERV1, INITC1, INITC2 und SERCT1

Neben den Routinen MAIN, SERV1 und INITC1 ist für diesen Versuch noch das folgende Programm notwendig:

Objekt-Code		Quell-Code	Bemerkung
		NAME INITC2	
FD21E006	INITC2:	LD IY,SERCT1	; Adresse der Service- Routine
FD22180F		LD (TABLE+18H),IY	; Tabelle einsetzen
3EC7		LD A,0C7H	; Steuerwort des Kanals 0
D310		OUT (10H),A	
3E01		LD A,01H	; Zeitkonstantenregister
D310		OUT (10H),A	; für Kanal 0
C3BD06		JP INITC1	;

Objekt-Code		Quelle-Code	Bemerkung
		NAME SERCT1	
C5	SERCT1:	PUSH BC	; BC-Registerinhalt zwi- ; schenspeichern
0E11		LD C,11H	; Port 11H von CTC
C33104		JP SERVI	

1. Schritt

Ergänzen Sie die Schaltung aus Bild 9-10 gemäß Bild 9-13.

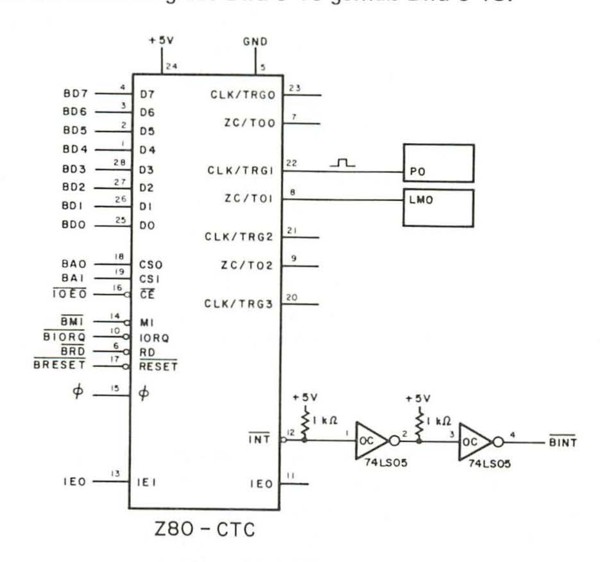


Bild 9-13. Zusatzanschlüsse für den CTC-Versuch Nr. 2.

Sie schließen eine externe Impulsquelle an den CLK/TRG-Eingang zum CTC-Kanal 0 an. Zum Triggern von Interrupt-Anforderungen von Kanal 0 zur Z-80-CPU den Impulsgeber P1 anschließen.

2. Schritt

Die einzige neue Software in diesem Versuch ist die Routine INITC2 mit folgenden Funktionen:

1. Adresse von SERCT1 in Vektortabelle laden
2. Steuerwort von Kanal 0 auf Hex-Ziffer C7 setzen (Es ist dasselbe, wie für Kanal 1 in diesem und Versuch 1).
3. Zeitkonstantenregister von Kanal 0 auf Hex-Ziffer 01 setzen.
4. Zu INITC1 springen.

INITC2 programmiert Kanal 0 mit der kleinstmöglichen Zeitkonstante, (nämlich 1) für den Zählbetrieb. SERCT1 ist die Interrupt-Service-Routine, die funktionell SERVID entspricht. Ihre Hauptfunktion besteht darin, den Inhalt eines festgelegten Eingang-Gatters zu lesen und anzuzeigen. An dem Codesegment SERVI, das von SERCT1 benutzt wird, muß folgende Änderung vorgenommen werden:

NOP (Hex-Ziffer 00) an Speicherstelle DSG+6H in EI-(Hex-Ziffer FB)-Befehl ändern.

Diese Änderung gibt Unterbrechungen während der Ausführung von SERCT1 frei, was zur Demonstration gewisser CTC-IC-Eigenschaften erforderlich ist.
Um die Wechselwirkung der verschiedenen Software-Elemente dieses Versuchs mit der Hardware zusammenzufassen, arbeitet der Impulsgeber P1 mit dem CTC-Kanal 0 zusammen; es entsteht so nach jedem Impuls eine Interrupt-Anforderung. Diese Unterbrechung bearbeitet die SERCT1-Routine, die den Inhalt des Rückwärtszählers von Kanal 1 liest und anzeigt. Der Rückwärtszähler des Kanals 1 wird durch Impulse des Impulsgebers P0 verändert. Nach fünf Impulsen von P0 erreicht der Rückwärtszähler von Kanal 1 den Nullzählzustand, woraufhin Kanal 1 $\overline{\text{BINT}}$ aktiviert. Die Interrupt-Service-Routine des Kanals 1 ist SERV1. Da es zwei mögliche Anforderer von CPU-Unterbrechungen gibt, nämlich die Kanäle 0 und 1, entscheidet die höhere Priorität. Was geschieht, wenn Kanal 1 eine Unterbrechung anfordert und Kanal 0 gerade eine Interrupt-Bearbeitung durchführt oder umgekehrt? Die Antwort liegt bei dem möglichen Steuerungsverlauf zwischen den sechs in diesem Versuch benutzten Routinen. Bild 9-14 zeigt die möglichen Steuerungsabläufe zwischen INITC1, INITC2,MAIN, SERCT1 und SERVI. Die folgenden Schritte befassen sich mit dem Prioritätsschema der CTC-Kanäle.
Bild 9-15 zeigt die Zeitfolge bei der von Routine SERCT1 ausgeführten CTC-Leseoperation.

3. Schritt

Ausführung bei INITC2 beginnen. Impulsgeber P1 betätigen.
Auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers® erscheint ca. 10 Sekunden lang 05. Der Rückwärtszähler von Kanal 1 hat zu Beginn die Zeitkonstante von Kanal 1 (05) geladen. Bisher noch keine Impulse am CLK/TRG1-Eingang. Der Stackpointer (Stapelzeiger) ist auf OEEC gesetzt.

4. Schritt

Impulsgeber P0 betätigen. Die Anzeige darf sich nicht ändern. Jetzt

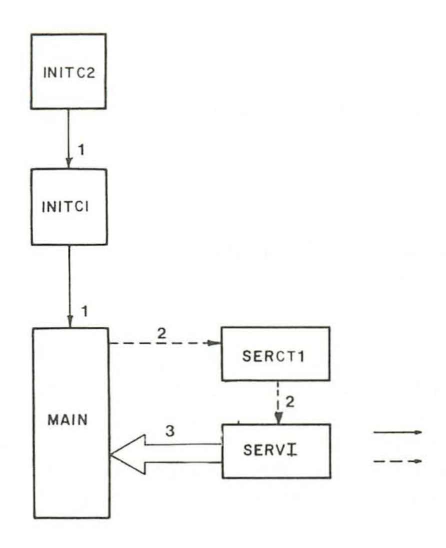


Bild 9-14. Möglichkeiten im Steuerungsablauf zwischen INITC1, INITC2, MAIN, SERCT1 und SERV1.
1 = Beendigung der INITC-Routinen
2 = Antwort auf Interrupt
3 = Rückkehr nach der Interrupt-Bearbeitung

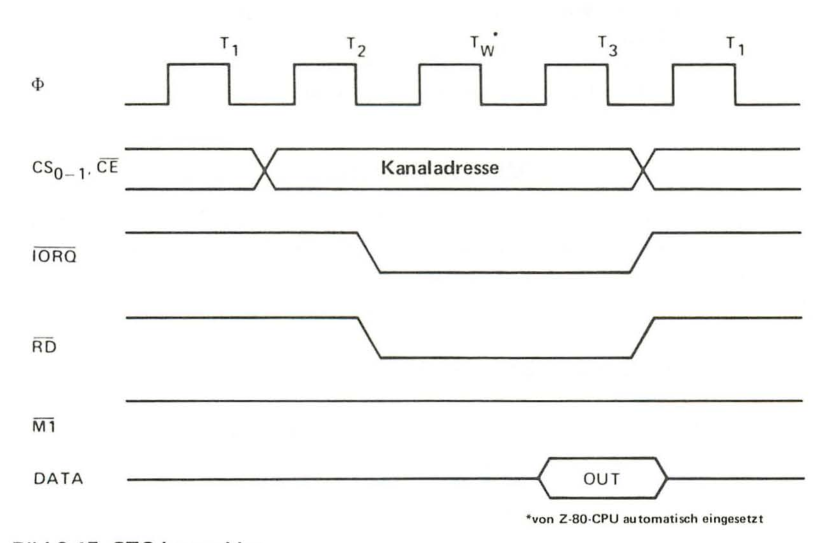


Bild 9-15. CTC-Lesezyklus

mit P1 einen Impuls auslösen. – Auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers® erscheint 04.

5. Schritt

Impulsgeber P0 dreimal auslösen. Die Anzeige ändert sich nicht. Impulsgeber P1 betätigen. Der Rückwärtszähler von Kanal 1 hat einen momentanen Inhalt von 01, den das Display anzeigt.

6. Schritt

Impulsgeber P0 noch einmal auslösen.
Es entsteht ein Interrupt und die Steuerung geht zur SERV1-Routine über. Die Stackpointer-Adresse ist OEEE.

7. Schritt

Während SERV1 abläuft, P1 betätigen. Die Ausführung von SERV1 hat Vorrang, um die Unterbrechung von Kanal 0 zu bearbeiten. Daher nimmt Kanal 0 eine höhere Prioritätsstellung in der Verkettung ein als Kanal 1. Der von SERCT1 für den Rückwärtszähler des Kanals 1 angezeigte Wert ist 5; das bedeutet, der Inhalt des Zeitkonstantenregisters wird nach Erreichen einer Nullzählung wieder automatisch in den Rückwärtszähler geladen.

8. Schritt

Der Stapelzeiger hat den Wert 0F00. Impulsgeber P1 auslösen. Während der folgenden Interrupt-Bearbeitung Impulsgeber P0 fünfmal direkt hintereinander betätigen.
Die Unterbrechung des Kanals 1 bleibt solange unerledigt, bis die Interrupt-Bearbeitung von Kanal 0 beendet ist. Erneut eine Bestätigung dafür, daß der Kanal 0 eine höhere Priorität hat als Kanal 1.
ANMERKUNG: Die komplette Prioritätsfolge für die CTC-Kanäle ist wie folgt: Kanal 0 > Kanal 1 > Kanal 2 > Kanal 3.

VERSUCH NR. 3

Er demonstriert den Betrieb eines CTC-Kanals nach der Zeittaktmethode. Durch Programmierung des Kanals 0 auf Zeittaktbetrieb und der anderen drei Kanäle auf Zählbetrieb können Sie die CTC als Stoppuhr umfunktionieren.

Programme: MAIN, INITC3 und SERCT2

Objekt-Code		Quell-Code	; Bemerkung
		NAME INITC3	
ED5E	INITC3:	IM2	; Interrupt-Methode Modus 2
21000F		LD HL,TABLE	; Vektor-Adreßtabelle
7C		LD A,H	; HI-Adreßbyte
ED47		LD I,A	; Interrupt-Register setzen
FD21E606		LD IY,SERCT2	; Interrupt-Service-Routine
FD22260F		LD (TABLE+26H),IY	; in Tabelle einsetzen
3E26		LD A,26H	; Interrupt-Vektor in

```
D314                 OUT (14H),A          ; CTC-Kanal 0 laden
08                   EX AF,AF'            ; Format für CONVDI setzen

3E40                 LD A,40H
08                   EX AF,AF'
3E2F                 LD A,2FH             ; Steuerwort Kanal 0
D314                 OUT (14H),A
3E96                 LD A,96H             ; Zeitkonstante Kanal 0

D314                 OUT (14H),A
3E47                 LD A,47H             ; Steuerwort Kanal 1
D315                 OUT (15H),A
3E40                 LD A,40H             ; Zeitkonstante Kanal 1

D315                 OUT (15H),A
3E47                 LD A,47H             ; Steuerwort Kanal 2
D316                 OUT (16H),A
3E00                 LD A,00H             ; Zeitkonstante Kanal 2

D316                 OUT (16H),A
3EC7                 LD A,0C7H            ; Steuerwort Kanal 3
D317                 OUT (17H),A
3E01                 LD A,01H             ; Zeitkonstante Kanal 3

D317                 OUT (17H),A
C3C302               JP MAIN
```

Objekt-Code		Quell-Code	Bemerkung

```
                     NAME SERCT2
C5        SERCT2:    PUSH BC              ; Inhalt der CPU-Register
D5                   PUSH DE              ; zwischenspeichern
E5                   PUSH HL
F5                   PUSH AF
DDE5                 PUSH IX
FDE5                 PUSH IY
FD2AE40F             LD IY,(ADDL)         ; Zustand von (ADDL) zwi-
FDE5                 PUSH IY              ; schenspeichern
0E16                 LD C,16H             ; Eingabe von CTC-Kanal 2
ED40                 IN B,(C)
AF                   XOR A                ; A zurücksetzen
90                   SUB B                ; Sekundenzahl suchen
32E40F               LD (ADDL),A          ; Daten von CTC-Kanal 2 in
                                          ; (ADDL) laden
DD23      DST:       INC IX               ; Stackpointer aktualisieren

DD23                 INC IX
DD23                 INC IX
00                   NOP                  ; keine Operation
DD3600FF             LD (IX+00H),0FFH     ; DLOOPT-Zeit setzen
DD36010A             LD (IX+01H),00AH     ; CLOOPT-Zeit setzen
DD360202  CLOOPT:    LD (IX+02H),02H      ; DLOOPT-Zeit setzen
21E50F               LD HL,ADDH           ; auf Anzeigepuffer hinweisen

ED57                 LD A,I               ; Wert von IFF2 feststellen
EA1C07               JP PE,HIGHT
```

```
3600        LOWT:    LD (HL),00H       ;Wert = 0
1802                 JR NEXTT
3610        HIGHT:   LD (HL),10H       ;Wert = 1
ED73E20F    NEXTT:   LD (DATA),SP      ; SP an Puffer übertragen
21B90F               LD HL,LEDL        ; für CONVDI setzen
11E50F               LD DE,ADDH        ; für CONVDI setzen
CD7CFA               CALL CONVDI
CD09F9      DLOOPT:  CALL DISPL
DD3500               DEC (IX+00)       ; Zeitgeber für Anzeige

20F8                 JR NZ,DLOOPT
DD3502               DEC (IX+02)       ; Zeitgeber für Anzeige

20F3                 JR NZ,DLOOPT
DD3501               DEC (IX+01)       ; Zeitgeber für Service-Routine

20CF                 JR NZ,CLOOPT
3E2F                 LD A,2FH          ; Steuerwort Kanal 0
D314                 OUT (14H),A
3E96                 LD A,96H          ; Zeitkonstante Kanal 0

D314                 OUT (14H),A
3E47                 LD A,47H          ; Steuerwort Kanal 1
D315                 OUT (15H),A
3E40                 LD A,40H          ; Zeitkonstante Kanal 1

D315                 OUT (15H),A
3E47                 LD A,47H          ; Steuerwort Kanal 2
D316                 OUT (16H),A
3E00                 LD A,00H          ; Zeitkonstante Kanal 2

D316                 OUT (16H),A
3EC7                 LD A,0C7H         ; Steuerwort Kanal 3
D317                 OUT (17H),A
3E01                 LD A,01H          ; Zeitkonstante Kanal 3

D317                 OUT (17H),A
FDE1                 POP IY            ; Inhalt von (ADDL) um-
                                         speichern
FD22E40F             LD (ADDL),IY
FDE1                 POP IY            ; Inhalte der Register in die
DDE1                 POP IX            ; CPU zurückgeben
F1                   POP AF
E1                   POP HL
D1                   POP DE
C1                   POP BC
FB                   EI                ; Interrupt-Flipflop freigeben

ED4D                 RETI              ; von Interrupt zurück
```

1. Schritt

Bauen Sie die in Bild 9-16 gezeigte Schaltung auf. Zwischen dieser und der Schaltung in den vorigen Versuchen bestehen hauptsächlich zwei Unterschiede. Der $\overline{CE}$-Eingang ist jetzt an IOE1 anstatt an IOE0 ange-

schlossen. Das hat hinsichtlich der Adressen der vier CTC-Kanäle folgende Auswirkungen:

Kanal	hexadezimale Adresse
1	14
2	15
3	16
4	17

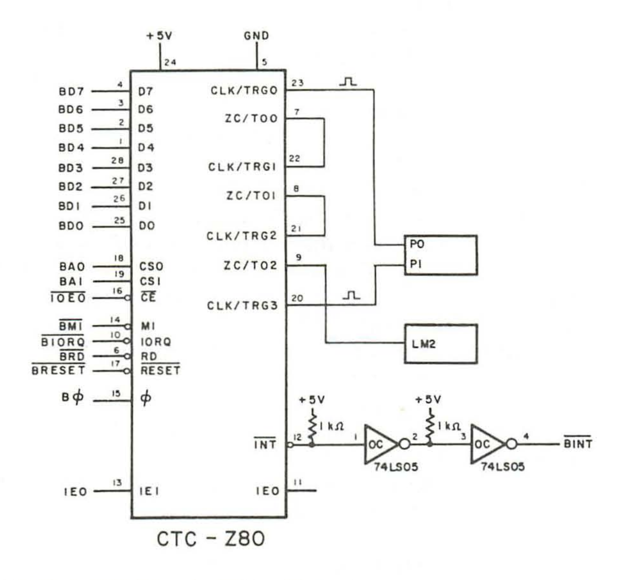


Bild 9-16. Schaltung für Zeittakt/Zähler-Versuch

Außerdem übernehmen die CLK/TRG- und ZC/TO-Leitungen verschiedene Aufgaben. Eine Nullzähler/Zeitsperre auf Kanal 0 taktet die CLK/TRG1-Leitung des Kanals 1. Wenn bei Kanal 1 eine Nullzähler/Zeitsperre-Situation eintritt, wird die CLK/TRG2-Eingabe über die ZC/T01-Leitung getaktet. Somit sind die Kanäle 0, 1 und 2 hintereinander geschaltet. Die ZC/T02-Leitung wird an LM2 angeschlossen. Der Impulsgeber P0 ist mit der CLK/TRG0-Leitung des Kanals 0 verbunden und der Impulsgeber P1 mit der CLK/TRG3-Leitung des Kanals 3.

2. Schritt

Neben dem MAIN-Programm benutzt dieser Versuch die Routinen INITC3 und SERCT2.

INITC3: Initialisierungs-Routine mit folgenden Funktionen:

1. Initialisierung der Interrupt-Methode zum Betrieb nach Modus 2.
2. Lokalisieren der Z-80-Interrupt-Vektortabelle an Speicherstelle TABLE.

3. Laden der SERCT2-Adresse Speicherstelle TABLE+26H.
4. Laden des CTC-Interrupt-Vektorregisters mit der Speicherstelle der Interrupt-Vektortabelle der SERCT2-Adresse.
5. Setzen des A'-Registers für CONVDI.
6. Byte 2F in das Steuerwort von Kanal 0 laden. Dieses Byte hat folgende Bedeutung:

Bit D7 = "0" – Kanal 0-Interrupts sind gesperrt
Bit D6 = "0" – Kanal 0 arbeitet nach Zeittaktmethode.
Bit D5 = "1" – Faktor des Impulsfrequenzteilers ist 256.
Bit D4 = "0" – Der Zeittaktbetrieb wird durch eine negative Flanke ausgelöst.
Bit D3 = "1" – Der Zeitgeber wird extern über einen Impuls auf der CLK/TRG0-Leitung getriggert.
Bit D2 = "1" – Das nächste zum Kanal 0 geschriebene Wort ist die Zeitkonstante.
Bit D1 = "1" – Nach Laden den Zeitkonstante soll Kanal 0 Betrieb wieder aufnehmen.
Bit D0 = "1" – Das Datenbyte ist das Kanal-Steuerwort.

7. Laden der Zeitkonstante für Kanal 0 mit Hex-Ziffer 96.
8. Byte 47 in Steuerwort von Kanal 1 laden. Dieses Byte veranlaßt Kanal 1 so zu arbeiten, wie in den Versuchen 1 und 2. Allerdings sind die CTC-Interrupts blockiert. Somit operiert der Kanal im Zählbetrieb. Wenn jedoch der Rückwärtszähler eine Nullzählung erreicht, erfolgt keine Interrupt-Anforderung. Eine Nullzählung erzeugt jedoch auf der ZC/T01-Leitung einen Impuls.
9. Laden der Zeitkonstante in Kanal 1 mit Hex-Ziffer 40.
10. Laden des Steuerworts in Kanal 2 mit dem Byte 47. Somit ist Kanal 2 genauso eingestellt wie Kanal 1.
11. Laden der Zeitkonstante in Kanal 2 mit der Hex-Ziffer 00. Bis zur Erreichung einer Nullzählung sind somit 256 Zählvorgänge notwendig.
12. Laden des Steuerworts in Kanal 3 mit dem Byte C7. Kanal 3 ist also wie Kanal 1 und 2 eingestellt, wobei allerdings CTC-Interrupts freigegeben sind.
13. Laden der Zeitkonstante in Kanal 3 mit Hex-Ziffer 01.

SERCT2: Diese Interrupt-Service-Routine bearbeitet eine von CTC-Kanal 3 angeforderte Unterbrechung, wenn eine Nullzählung erreicht ist. Sie hat folgende Funktionen:

1. Zwischenspeichern der Registerinhalte auf dem Stapelspeicher.
2. Zwischenspeichern des vom Programm MAIN angezeigten Zählerzustands.
3. Eingabe des Rückwärtszähler-Inhaltes von Kanal 2.
4. Da jede Verminderung des Rückwärtszählers in Kanal 2 bei einem Ausgangswert von 00 einer tatsächlichen Zeiteinheit von 1 Sekunde entspricht, ist die Differenz zwischen 00 und dem laufenden Wert des Rückwärtszählers gleich der verstrichenen Sekundenzahl. Diese Differenz berechnet der Befehl XOR A
   SUB B
5. Anzeige der in der ADDL-Position verstrichenen Sekundenzahl.
6. Anzeige des laufenden Wertes von IFF2, der verstrichenen Sekunden-

zahl und des Stapelzeigers.

7. Rücksetzen der Stoppuhr durch erneutes Initialisieren der Zeitkonstanten für alle CTC-Kanäle.
8. Umspeichern der CPU-Register.
9. Freigabe von Interrupts und RETI.

3. Schritt

Dieser Schritt behandelt die Grundsätze von Soft- und Hardware für diesen Versuch. Indem die Ausführung der Software an der Speicherstelle INITC3 beginnt, werden die vier Kanäle des CTC-ICs wie beschrieben programmiert. Kanal 0 arbeitet nach der Zeittaktmethode, wobei die Zeitfolge durch einen Impuls vom Impulsgeber P0 bestimmt wird. Das Schalten von P0 löst also den Zeittaktbetrieb aus. Der Rückwärtszähler (Kanal 0) erreicht ca. 1/64 Sekunde später einen Nullzählstand und taktet die CLK/TRG1-Leitung, wodurch der Rückwärtszähler von Kanal 1 um 1 dekrementiert. Kanal 0 schickt weiterhin alle 1/64 Sekunde einen Impuls zur CLK/TRG1-Leitung ($64_{10} = 49_{16}$). Nach 64 solcher Impulse ist eine Sekunde verstrichen. Der Rückwärtszähler von Kanal 1 erreicht einen Nullzählerstand. Das hat einen Impuls an die CLK/TRG2-Leitung zur Folge, so daß der Rückwärtszähler von Kanal 2 ebenfalls um 1 dekrementiert. Da der Rückwärtszähler von Kanal 2 auf 00 eingestellt ist, registriert er das *Gegenteil* der verstrichenen Sekundenzahl. Die Interrupt-Service-Routine SERCT2 macht sich dies zunutze und errechnet die richtige anzuzeigende Zahl als verstrichene Zeit in Sekunden. Die von der Stoppuhr maximal zu messende und angezeigte Zeit beträgt 255 Sekunden.

Da Kanal 3 als Zähler mit einer Zeitkonstanten = 1 fungiert, sein Interrupt freigegeben und SERCT2 die Service-Routine ist, wird durch Betätigen des Impulsgebers P1 die verstrichene Zeit in Sekunden auf der Tastenfeld-/Anzeigeeinheit des Nanocomputers® angezeigt.

4. Schritt

Probieren Sie Ihren Zeitgeber aus, indem Sie das Programm bei INITC3 starten. Messen Sie die zwischen der Auslösung des Impulsgebers P0 und P1 verstrichene Zeit mit Hilfe einer externen Stoppuhr. Nach Prüfung mehrerer Zeitintervalle stellen Sie fest, daß die CTC-Stoppuhr sehr genau arbeitet.

Anhang **A**

Beschreibung der in den Versuchen benutzten Software

In diesem Anhang finden Sie zusätzliche Angaben zu den in den Versuchen benutzten Routinen. Zunächst eine Erörterung der allgemeinen Speicherorganisation. Die folgende Aufstellung ist ein höherer Speicherplan der 4K-Bytes des Lese-/Schreibspeichers.

Speicherplan für die Software-Zuteilung 2.2 bei den Nanocomputer®-Versuchen

Adresse	Inhalt	
0000–00FF	RAM SEITE (246 BYTES)	
0100–07BC	VERSUCHSPROGRAMM ABLAUFBEREITER CODE	
0800–0BFF	LISTE FÜR CHPTST AUFFÜLLEN	
0C00 –	DSTACK ANFANGSWERT DES DATEN STACK POINTER ↓ ↑	UNKIC-LISTE FÜR CHPTST
0EFF –	Z-80-STACK POINTER ANFANGSWERT VON SP IST 0F00 FÜR BETRIEBSSYSTEM DES NANOCOMPUTERS® UND ALLE VERSUCHSROUTINEN AUSSER CHPTST	
0F00–0F30	INTERRUPT-VEKTORTABELLE BENUTZT VON VERSUCHSROUTINEN AUSSER CHPTST ↑ ↓	
0FA0	ANFANGSWERT DES STACK POINTER FÜR CHPTST	
0FAB–0FFF	RAM-SPEICHERSTELLE BENUTZT VOM NANOCOMPUTER®-BETRIEBSSYSTEM UND IN DEN VERSUCHEN	

Der niedrigwertigste 256-Byte-Block des Lese-/Schreibspeichers ist in der Regel die Seite 0. Sie wird vom Z-80-Befehlsgerät, zwei der Z-80-Interrupt-Methoden und dem Betriebssystem des Nanocomputers® benutzt. Für die Seite 0 gilt folgender Speicherplan.

Seite 0 des RAMs (256 Bytes)

0000H–RST 00H
0008H–RST 08H
0010H–RST 10H
0018H–RST 18H
0020H–RST 20H
0028H–RST 28H
0030H–RST 30H
0038H–RST 38H–Z-80 Interrupt-Methode 1
0066H–Z-80 Nichtmaskierbarer Interrupt
00FFH

Alle acht RESTART-Befehle benutzen Seite 0. Außerdem ist die Antwort auf ein aktives $\overline{INT}$-Signal für die Z-80-Interrupt-Methode 1 funktionell äquivalent mit einem Wiederanlauf bei Speicherstelle 0038 und 0066. Das Betriebssystem des Nanocomputers® benutzt Seite 0, um sich selbst wieder einzubeziehen, nachdem man die RESET-Taste betätigt hat. Deshalb ist im Umgang mit Seite 0 entsprechende Vorsicht geboten.
Der ablaufbereite Code für die Versuche befindet sich an der nächsten Speicherstelle, 0100. Die Master-Symbol-Tabelle (Symbol-Rahmentabelle) für diesen Code ist in Tabelle A-1 dargestellt. Jede Symboladresse in der Tabelle verweist auf ihre absolute Speicheradresse. Tabelle A-2 ist eine Symbolauflistung mit Querverweisen. Alle vom Programm benutzten Symbole sind in alphabetischer Reihenfolge aufgeführt und mit einer oder mehreren Zahlen versehen. Es handelt sich hierbei um die fortlaufende Numerierung der Speicherstellen. Das komplette Programm der Versuchs-Software mit absoluten Speicheradressen und fortlaufender Numerierung ist in Anhang B aufgeführt. Die Software für Kapitel 1 bis 5 ist relativ problemlos. Die folgende Erörterung befaßt sich deshalb mit den restlichen Software-Routinen. Der Speicherplatz von INITO bis DSTACK (wobei DSTACK der Anfangswert des Daten-Stackpointers ist) enhält *nur* ablaufbereite Codes und keine Daten-Speicherplätze. Tabelle A-3 ist ein detaillierter Plan dieser Speicherplätze. Die Bezeichnungen beziehen sich auf die Startadressen der ablaufbereiten Codes. Zwischen den Routinen in diesen Versuchen und den Eintrittspunkten in den ablaufbereiten Codes besteht eine hundertprozentige Übereinstimmung.
Auf den Betrieb des Daten-Stackpointers geht dieser Abschnitt an anderer Stelle ein. Zunächst nur soviel: Der Anfangswert des Daten-Stackpointers befindet sich auf dem ablaufbereiten Code (d.h. an einer höheren Adressen-Speicherstelle des Speichers) und der Stackpointer nimmt im Speicher nach oben zu, d.h. in Richtung höherer Speicherstellen.
Das Betriebssystem des Nanocomputers® initialisiert den Stackpointer auf 0F00. Das bedeutet, das erste auf den Stack geschobene Byte besetzt die Speicherstelle 0EFF bis 0EFE. Der Stackpointer dehnt sich auf nacheinanderfolgende niedrige Speicherstellen aus. Dadurch besteht die Gefahr, daß Datenspeicher und Stackpointer ineinander übergehen.
ANMERKUNG: In der Praxis sollte man über Begrenzungs-Prüfroutinen verfügen, welche die Bewegung beider Register kontrolliert.

Tableau A-1. Master Symbol Tabelle

Symbol	Absolute Speicher-adresse	Symbol	Absolute Speicher-adresse	Symbol	Absolute Speicher-adresse	Symbol	Absolute Speicher-adresse	Symbol	Absolute Speicher-adresse
ADD7	0FBA	ADDH	0FE5	ADDL	0FE4	BAD	06B9	BAUD	F9F2
BAUDRT	0FAE	BLKMVE	F000	CHECK	012F	CHECKB	F99D	CHPSTK	0FA0
CHPTST	0647	CLOOP1	0285	CLOOP3	0382	CLOOPG	0453	CLOOPM	0531
CLOOPN	0330	CLOOPT	070C	CLOOPX	05FE	COMPAR	0677	CONVDI	FA7C
CWORD	04F4	DATAH	0FE3	DATAL	0FE2	DATALP	01CA	DDRIVE	01B6
DECODE	010A	DELAY	01E3	DISAB	02E6	DISPL	F909	DISTST	01C7
DLOOP	02EA	DLOOP1	02A6	DLOOP3	03A4	DLOOPG	0472	DLOOPM	0550
DLOOPN	034F	DLOOPT	072B	DLOOPX	061F	DREGL	01E6	DS	F02D
DS1	0276	DS3	0373	DSG	0444	DSM	0522	DSN	0321
DSPLAY	0208	DST	06FD	DSTACK	0C00	DSX	05EF	ENABG	044A
END	0164	ENDREF	0669	ERRLP	015F	ERROR	0144	GETNO	01F3
GOOD	0686	ENPIO	03DD	HIGH1	0295	HIGH3	0392	HIGHG	0463
HIGHM	0541	HIGH	02D8	HIGHT	071C	HIGHX	060E	INITO	021B
INIT1	0231	HIGHN	0340	INIT2	0247	INITC1	06BD	INITC2	07A9
INITC3	076E	INIT1N	02F6	INITID	03F4	INITOC	03C1	INITPB	0495
INITPM	04D4	INITDC	05AA	KBSCAN	F8DB	KBTST	01EE	LEDH	0FB8
LEDL	0FB9	INITPP	0573	LOOP1	0100	LOOP2	0104	LOOP3	010E
LOOP4	0120	LENGHT	0700	LOOP6	01A0	LOW	02D4	LOW1	0291
LOW3	038E	LOOP5	019D	LOWM	053D	LOWN	033C	LOWT	0718
LOWX	060A	LOWG	045F	MASK	0694	MASKW	0003	MEM1	011E
MOVE	F014	MAIN	02C3	NEXT	02DA	NEXT1	0297	NEXT3	0394
NEXTB	067D	NANOR2	F00D	NEXTM	0543	NEXTN	0342	NEXTT	071E
NEXTX	0610	NEXTG	0465	NTEST	068B	NXTLOC	0135	NXTWD	06B3
OK	017F	NEXXT	013E	ORIGIN	0100	OUTPUT	01CD	OUTSIM	0212
OUTX	0637	ORGIN	0100	PSEL	0000	PULSR	0112	REF	065C
REFIC	0800	PCNTR	01AD	SERCT1	06E0	SERCT2	06E6	SEROCX	05E1
SERV1	026E	RESTAR	F08E	SERV3	036B	SERVI	0431	SERVIC	0419
SERVID	041F	SERV2	02F5	SERVIF	042B	SERVM	0505	SERVN	0319
SERVOC	03E8	SERVIE	0425	STORE	0688	STRING	F042	TABLE	0F00
TEST	0698	START	06BB	UCINM	01AB	UCINP	0190	UNKIC	0C00
UNKN	066A	THROW	03E3	XFER	0184				
		WAIT	019A						

Tabelle A-2. Komplette Hinweisliste auf die durchnumerierten Speicherplätze der einzelnen Routinen.

ADD7	0032	0367						
ADDH	0021	0153	0162	0171	0176	0366	0450	0460
		0484	0494	0540	0548	0582	0593	0714
		0722	0828	0836	0930	0940	1136	1144
ADDL	0020	0172	0638	0703	0706	0732	0810	0815
		0846	0921	0950	0951	1122	1128	1170
BAD	1079	1016						
BAUD	0037	0319						
BAUDRT	0029							
BLKMVE	1236							
CHECK	0123							
CHECKB	0030	0340	0372					
CHPSTK	0038	0983	0994					
CHPTST	0965							
CLOOP1	0449	0468						
CLOOP3	0581	0601						
CLOOPG	0713	0730						
CLOOPM	0827	0844						
CLOOPN	0539	0556						
CLOOPT	1135	1152						
CLOOPX	0929	0948						
COMPAR	1005							
CONVDI	0027	0163	0177	0368	0461	0496	0549	0594
		0723	0837	0941	1145			
CWORD	0792							
DATAH	0023	0160	0173					
DATAL	0024	0158	0174	0353	0458	0492	0546	0591
		0720	0834	0938	1142			
DATALP	0281	0314						
DDRIVE	0243							
DECODE	0065							
DELAY	0317	0300						
DISAB	0495							
DISPL	0028	0164	0178	0370	0462	0497	0550	0595
		0724	0838	0942	1146	1267		
DISTST	0277							
DLOOP	0497	0499						
DLOOP1	0462	0464	0466					
DLOOP3	0595	0597	0599					
DLOOPG	0724	0726	0728					
DLOOPM	0838	0840	0842					
DLOOPN	0550	0552	0554					
DLOOPT	1146	1148	1150					
DLOOPX	0942	0944	0946					
DREGL	0319	0333						
DS	1267	1269	1271					
DS1	0443							
DS3	0575							
DSG	0707							
DSM	0821							
DSN	0533							
DSPLAY	0370	0374						
DST	1129							

DSTACK 0036 0482 1249
DSX 0923
ENABG 0710
END 0167 0142
ENPIO 0627
ENDREF 0992
ERRLP 0164 0165
ERROR 0145 0129
GETNO 0345 0376
GOOD 1016
HIGH 0489 0486
HIGH1 0455 0452
HIGH3 0587 0584
HIGHG 0719 0716
HIGHM 0833 0830
HIGHN 0545 0542
HIGHT 1141 1138
HIGHX 0935 0932
INIT0 0393
INIT1 0406
INITIN 0510
INIT2 0419
INITC1 1089 1221
INITC2 1215
INITC3 1183
INITDC 0887
INITID 0648
INITOC 0614
INITPB 0745
INITPM 0777
INITPP 0859 0910
KBSCAN 0031 0345
KBTST 0340 0342 0350
LEDH 0025 1252 1264 1266
LEDL 0026 0161 0175 0459 0493 0547 0592 0721 0835 0939 1143
LENGHT 1232 1240
LOOP1 0046 0049 1311
LOOP2 0055 0059
LOOP3 0075 0079
LOOP4 0109 0140
LOOP5 0215 0223
LOOP6 0216 0219
LOW 0487
LOW1 0453
LOW3 0585
LOWG 0717
LOWM 0831
LOWN 0543
LOWT 1139
LOWX 0933
MAIN 0481 0401 0414 0432 0500 0522 0631 0664 0772 0799 0882 1104 1210
MASK 1039
MASKW 0033 0969 1029 1051
MEM1 0107 0184 0189

MOVE 1252 1281
NANOR2 1249
NEXT 0490 0488
NEXT1 0456 0454
NEXT3 0588 0586
NEXTB 1012 1020
NEXTG 0720 0718
NEXTM 0834 0832
NEXTN 0546 0544
NEXTT 1142 1140
NEXTX 0936 0934
NEXXT 0139 0131
NTEST 1029 1071
NXTLOC 0127 0135
NXTWD 1067 1042
OK 0178 0179 0184
ORGIN 0017 0018
ORIGIN 1231 1239
OUTPUT 0286 0305
OUTSIM 0382
OUTX 0951
PCNTR 0231 0237
PSEL 0022 0251
PULSR 0086
REF 0983
REFIC 0034 0985 1005
RESTAR 1307 1237
SERCT1 1109 1215
SERCT2 1116 1187
SEROCX 0915
SERV1 0437 0395 0408 0423 0517 1093
SERV2 0505 0425
SERV3 0569 0427
SERVI 0697 0671 0678 0685 0692 1111
SERVIC 0669 0751 0863
SERVID 0676 0652 0865
SERVIE 0683 0891
SERVIF 0690 0893
SERVM 0804 0781
SERVN 0527 0512
SERVOC 0636 0618 0749
START 1082 1022 1079
STORE 1027 0990 1001
STRING 1288 1251
TABLE 0019 0089 0420 0424 0426 0428 0615 0619 0649 0653 0746 0750 0752 0778 0782 0860 0864 0866 0888 0892 0894 1090 1094 1184 1188 1216
TEST 1045
THROW 0630
UCINM 0229
UCINP 0200
UNKIC 0035 0996 1008
UNKN 0994 1082
WAIT 0214 0202 0235
XFER 0184

Tabelle A-3. Startadressen der Software-Routinen bei den Nanocomputer®-Versuchen.

Start-adresse	Routine
0100	LOOP1
0104	LOOP2
010A	DECODE
0112	PULSR
011E	MEM1
0184	XFER
0190	UCINP
01AB	UCINM
01B6	DDRIVE
01C7	DISTST
01EE	KBTST
0212	OUTSIM
021B	INIT0
0231	INIT1
0247	INIT2
026E	SERV1
02C3	MAIN
02F5	SERV2
02F6	INIT1N
0319	SERVN
036B	SERV3
03C1	INITOC
03E8	SERVOC
03F4	INITID
0419	SERVIC
041F	SERVID
0425	SERVIE
042B	SERVIF
0431	SERVI
0495	INITPB
04D4	INITPM
0505	SERVM
0573	INITPP
05AA	INITDC
05E1	SEROCX
0647	CHPTST
06BD	INITC1
06E0	SERCT1
06E6	SERCT2
076E	INITC3
07A9	INITC2
F000	BLKMVE
F00D	NANOR2

Nimmt der Stackpointer zwei neue Bytes auf, sollte eine Prüf-Routine darauf achten, daß der Datenspeicher nicht berührt wird. Andererseits kann die Prüf-Routine bei der Byte-Entnahme aus dem Stackpointer verhindern, daß man die Anfangsadresse 0F00 überschreitet.

Die Vektortabelle befindet sich im Speicher ab 0F00. Eine detailliierte Beschreibung der Einträge in die Vektortabelle zeigt Tabelle A-4.

Tabelle A-4. Startadressen der Vektortabellen.

Adresse	Eintrag
0EFF –	Z-80 STACK POINTER (Stapelzeiger) (Anfangswert von SP = 0F00)
0F00 –	SERV1 (LO-Adreß-Byte)
1	SERV1 (HI-Adreß-Byte)
2	SERV2
3	SERV2
4	SERV3
5	SERV3
6	SERV0C
7	SERV0C
8	SERVID
9	SERVID
A	SERVIC
B	SERVIC
C	SERVM
D	SERVM
E	SERVIE
F	SERVIE
0F10	SERVIF
11	SERVIF
0F18	SERCT1
0F19	SERCT1
0F1A	SERV1
0F1B	SERV1
0F26	SERCT2
0F27	SERCT2
0FB9	LEDL
0FE2	DATAL
3	DATAH
4	ADDL
5	ADDH
0FFF	

Schließlich werden die letzten 84 Bytes (höchstwertig) des Lese-/Schreibspeichers vom Betriebssystem des Nanocomputers® benutzt. In diesen 84 Bytes gibt es vier spezifische Stellen, auf welche die Versuchsroutine wiederholt Bezug nimmt. Diese Stellen sind in Tabelle A-4 aufgeführt. Die Zuordnung zwischen diesen Stellen, den Bezeichnungen und dem Display

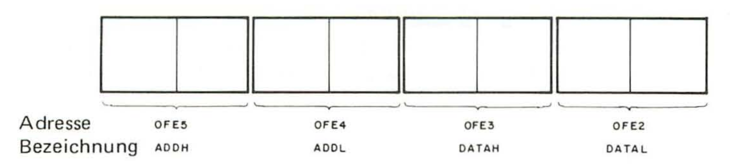


Bild A-1. Interne Pufferanordnung des Nanocomputer®-Betriebssystems zum Betreiben der Tastenfeldanzeige.

des Tastenfeldes, das den Wert anzeigt, ist in Bild A-1 dargestellt.
Tabelle A-5 identifiziert jede der 8 PIO-Portadressen sowie die CTC-Portadresse, auf welche die Routinen mit dem entsprechenden Portnamen Bezug nehmen.

Tabelle A-5. PIO-I/O-Adresse

PIO	Portname	Type	Port Adresse
PIO N° 2	Port C	Datenadresse	08H
PIO N° 2	Port D	Datenadresse	09H
PIO N° 2	Port C	Steueradresse	0AH
PIO N° 2	Port D	Steueradresse	0BH
PIO N° 3	Port E	Datenadresse	0CH
PIO N° 3	Port F	Datenadresse	0DH
PIO N° 3	Port E	Steueradresse	0EH
PIO N° 3	Port F	Steueradresse	0FH

PIO2 gehört zum Nanocomputer® und ist für den Anwender über die Experimentier-Platine zugänglich. PIO3 steht dem Anwender über die Experimentier-Platine zur freien Verfügung.

Die Routinen in den Kapiteln 6, 7 und 9 lassen sich in drei Hauptgruppen unterteilen:

Gruppe 1: Initialisierungs-Routine

INITO, INIT1, INIT2, INITIN, INITOC, INITID, INITPB, INITPP, INITDC, INITC1, INITC2, INITC3.

Gruppe 2: Die Hauptbearbeitungsroutine MAIN.

Gruppe 3: Die Interrupt-Service-Routinen:

SERV1, SERV2, SERV3, SERVN, SERVOC, SERVID, SERVM, SERVIC, SERVIE, SERVIF, SERCT1, SERCT2, SEROCX, SERVI.

Gruppe 1

Die Initialisierungs-Routinen sind problemlos, daher werden nur die Befehlsfolgen

```
EX AF,AF'
LD A,40H
EX AF,AF'
```

erklärt.
Diese Befehlsfolge hat den Zweck, das Format für die Leerstellenanzeige der Subroutine CONVDI zu setzen. Die Subroutine CONVDI benutzt das Wechselregister A'. Sie bestimmt, welche der Tastenfeld-Sieben-Segment-Anzeigen aufleuchten und welche dunkel bleiben sollen.

Soll eine vorgegebene Sieben-Segment-Anzeige aktiv sein, muß das entsprechende Bit des A'-Registers eine 0 enthalten. Bild A-2 zeigt das Format der Standardanzeige, die das Buch benutzt und die stets nur das zweite Anzeige-Segment von links dunkel läßt.

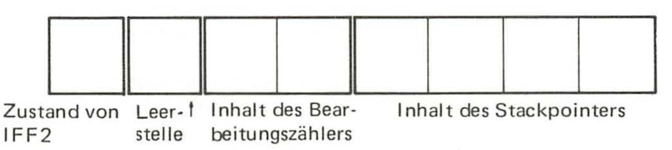


Bild A-2. Format der Tastenfeld-Anzeige bei Nanocomputer®-Versuchen.

Daher ist der zur Erzeugung der Anzeige benutzte Wert des A'-Registers immer 0 1 0 0 0 0 0 0 oder 40 (Hex.-Ziffer). Das ist der Grund dafür, warum die Initialisierungs-Routinen die obige Folge der drei Anweisungen enthält.

Gruppe 2

Die MAIN-Routine ist in vielen Funktionen denen der Programmunterbrechungen ähnlich:

1. Zustandsanzeige des Interrupt-Flipflops IFF2 auf der linksbündigen Stelle der Tastenfeld-Anzeigeeinheit.
2. Inhaltsanzeige des Stackpointers auf den vier rechtsbündigen Stellen.
3. Verminderung (Dekrementierung) des Bearbeitungszählers ADDL.
4. Anzeige von ADDL.

Zur Ausübung dieser Funktionen lädt die MAIN-Routine den 4-Byte-Anzeigepuffer mit den bei SAMPLE TIME erhaltenen Werten. Bilder A-1 und A-2 zeigen die Zuordnung zwischen den Anzeige-Pufferadressen, den für die Anzeigepuffer im Betriebssystem des Nanocomputers® benutzten Bezeichnungen und der Anzeigestellung, die ihnen jeweils zugeteilt sind.
Die Funktionen der MAIN-Routinen sind:

```
MAIN:   EI                  ; Interrupt-Flipflop IFF1 frei-
                            ; geben
        LD IX,DSTACK        ; IX-Register auf den DSTACK-
                            ; Wert initialisieren, der den
                            ; unteren Teil des Stackpointers darstellt.
        LD (IX+00),0FFH     ; das erste Element im Stack-
                            ; pointer auf FFH initialisieren.
                            ; Das erste Element legt fest,
```

```
                                ;wie oft die Subroutine DISPLY
                                ;abläuft, um einen Wert in ADDL
                                ;anzuzeigen. Bei dem Wert FFH
                                ;dauert die Anzeige ca. 0,5 Se-
                                ;kunden.
        LD HL,ADDH              ;HL initialisieren, um auf
                                ;den Puffer hinzuweisen, der
                                ;den Wert von IFF2 enthält.
        LD A,I                  ;Diese Befehlsfolge enthält den
                                ;Wert von IFF2 und schreibt ihn
                                ;in den Puffer ADDH. Die Anwei-
        JP PE,HIGH              ;sung JP PE,HIGH ist die ein-
LOW:    LD (HL),00H             ;fachste Methode, um den Wert
        JR NEXT                 ;von IFF2 zu bestimmen und
                                ;festzu-
HIGH:   LD (HL),10H             ;halten.
NEXT:   DEC HL                  ;HL = ADDL setzen, damit HL auf
                                ;den Bearbeitungszähler hin-
                                ;weist.
        DEC (HL)                ;den Bearbeitungszähler (ADDL)
                                ;dekrementrieren.
                                ;Stackpointerinhalt zur
        LD (DATAL),SP           ;anschließenden Anzeige in den
                                ;Puffer laden.
        LD HL,LEDL              ;Abruf der Subroutine CONVDI
                                ;vorbereiten.
        LD DE,ADDH
DISAB:  NOP                     ;Dieses Byte steht bereit, um
                                ;bei den Versuchen als indivi-
                                ;dueller 1-Byte-Befehl zur Ver-
                                ;fügung zu stehen.
        CALL CONVDI             ;diese Befehlsfolge zeigt die
                                ;abgetasteten Parameter an.

DLOOP:  CALL DISPLY
        DEC (IX+00H)
        JR NZ,DLOOP
        JP MAIN                 ;Rücksprung und die Parame-
                                ;ter erneut abtasten.
```

Jede Versuchs-Routine benutzt das IX-Register als globale Variable; es enthält immer den momentanen Wert des "Daten-Stackpointers". Register IX ist also insofern global, als jede Routine das Register für genau denselben Zweck benutzt und das Register für jede Routine dieselbe Bedeutung hat.

Eine genaue Operationsbeschreibung des Daten-Stapelanzeigers (Daten-Stackpointer) ist in der Beschreibung der Routine SERV1 enthalten. Um die MAIN-Routine zu verstehen, muß man nur erkennen: LD IX,DSTACK initialisiert den Wert dieses Stapelzeigers zu DSTACK. Im Falle der MAIN-Routine wird nur ein Byte vom Datenstapel benutzt (IX + 00H) = (DSTACK + 00H). Dieses Byte hält den Wert des Zeitgebers für die Anzeigeschleife DLOOP konstant. Die Zeitgeber-Konstante wird zu FFH initialisiert. Die Befehlsfolge

```
          LD HL,ADDH
          LD A,I
          JP PE,HIGH
LOW:      LD (HL),00H
          JR NEXT
HIGH:     LD (HL),10H
NEXT:
```

tastet den Wert von IFF2 ab und setzt die IFF2-Anzeigestelle in ADDH auf "0", falls IFF1 = "1" ist. Dabei ist der Wert von IFF2 ("0" oder "1") irrelevant. Dies ist möglich, weil der Zustand von IFF2 in das Paritäts-Flag wechselt, wenn der Befehl LD A,I ausgeführt wird. Wie bekannt, entspricht die gerade Parität P = "1" und die ungerade Parität P = "0". Wenn also P = "1" ist, wird die Anweisung LD(HL),10H nicht ausgeführt; das höherwertige Halbbyte von ADDH bleibt = 0. Das höherwertige Halbbyte von ADDH entspricht der für IFF2 angezeigten Ziffer.
Der Bearbeitungszähler arbeitet mit der Befehlsfolge

```
DEC HL
DEC (HL)
```

Da HL initialisiert, um auf ADDH = 0FEF hinzuweisen, weist DEC HL auf 0FE4 = ADDL hin, dem Puffer für den Bearbeitungszähler.
LD(DATA),SP tastet den Stackpointer ab und lädt den Inhalt in einen 2-Byte-Speicherpuffer. Die Anweisungen

```
LD HL,LEDL
LD DE,ADDH
```

stellen die für die Subroutine CONVDI erforderliche Vorbereitung dar, die Teil des Nanocomputer®-Betriebssystems ist. Die Anzeige wird durch wiederholte Ausführungen der Routine DISPLY (Nanocomputer®-Betriebssystem) herbeigeführt.

Gruppe 3
Die Interrupt-Service-Routine SERV1 ist typisch für diese Software-Gruppe. Sie hat folgende Funktionen:

1. Zwischenspeichern der CPU-Register-Inhalte.
2. Aktualisierung des Daten-Stackpointers.
3. Initialisierung der Zeitgeber-Puffer auf dem Datenstapel.
4. Übernahme des momentanen Zustandes von Interrupt-Flipflop IFF2 zur Anzeigenvorbereitung.
5. Handhabung des Bearbeitungszählers bei ADDL.
6. Übernahme des momentanen Stackpointer-Inhaltes zur Anzeigevorbereitung.
7. Anzeige des Stapelzeigers und IFF2 – zusätzlich zur Anzeige des laufenden Werts vom Bearbeitungszähler.
8. Handhabung einer Zeitgeberschleife, die 10 Wiederholungen durchläuft, abtastet und anzeigt.
9. Umspeichern der CPU-Register-Inhalte

OPERATION DES DATEN-STACKPOINTERS

Der Daten-Stackpointer arbeitet ähnlich wie der Z-80-Stapelzeiger. Der untere Teil des Stapels ist der Anfangswert des Daten-Stapelzeigers,

nämlich DSTACK. Dieser Stapelspeicher wächst nach oben – im Gegensatz zum Z-80-Stapelspeicher, der nach unten wächst. Der Daten-Stapelzeiger wird initialisiert, damit er beim Wachsen weder den ablaufbereiten Code noch die Tabellen anderer überschreibt. Der Speicher-Rahmenplan liefert ein Bild der Speicherstelle von DSTACK, dem Unterteil des Stapels, im Verhältnis zu den Speicherstellen der anderen Puffer und Codes.
Jede Routine, die den Datenstapel benutzt, muß eingetragen werden. Sie ist dem Daten-Stackpointer für jedes Daten-Byte, das sie benötigt, um eines Speicheradresse voraus. Versetzt eine Routine den Daten-Stack-Pointer, speichert sie auch den von ihr benutzten Wert des unteren Datenstapels und übernimmt ihn nach der Bearbeitung in das IX-Register.

Anhang **B**

Komplettes Listing der Versuchs-Software

Die Kommentare im Programm-Listing sind in Englisch. Die deutschen Kommentare finden Sie in den entsprechenden Programmen der jeweiligen Versuche.

```
        0001  ;
        0002  ;
        0003  ;
        0004  ;
        0005  ;
        0006  ;
        0007  ;
        0008  ;
        0009  ;
        0010  ;
        0011  ;
        0012  ;
        0013  ;
        0014  ;
        0015  ;
        0016            NAME REL2.2
(0100)  0017  ORGIN     EQU 0100H
        0018            ORG ORGIN
(0F00)  0019  TABLE     EQU 0F00H
(0FE4)  0020  ADDL      EQU 0FE4H
(0FE5)  0021  ADDH      EQU 0FE5H
(0000)  0022  PSEL      EQU 00H
(0FE3)  0023  DATAH     EQU 0FE3H
(0FE2)  0024  DATAL     EQU 0FE2H
(0FB8)  0025  LEDH      EQU 0FB8H
(0FB9)  0026  LEDL      EQU 0FB9H
(FA7C)  0027  CONVDI    EQU 0FA7CH
(F909)  0028  DISPL     EQU 0F909H
(0FAE)  0029  BAUDRT    EQU 0FAEH
(F99D)  0030  CHECKB    EQU 0F99DH
(F8DB)  0031  KBSCAN    EQU 0F8DBH
```

```
        (0FBA)   0032  ADD7     EQU 0FBAH
        (0003)   0033  MASKW    EQU 0003H
        (0800)   0034  REFIC    EQU 0800H
        (0C00)   0035  UNKIC    EQU 0C00H
        (0C00)   0036  DSTACK   EQU 0C00H
        (F9F2)   0037  BAUD     EQU 0F9F2H
        (0FA0)   0038  CHPSTK   EQU 0FA0H
                 0039  ;
                 0040  ;
                 0041  ;
                 0042  ;
                 0043  ;
                 0044  ;
                 0045           NAME LOOP1
0100    D3C5     0046  LOOP1:   OUT (0C5H),A    ;Output the contents
                 0047                           ;of the accumulator
                 0048                           ;to port C5
0102    18FC     0049           JR LOOP1        ;Repeat until break
                 0050  ;                        ;or reset
                 0051  ;
                 0052  ;
                 0053  ;
                 0054           NAME LOOP2
0104    3E21     0055  LOOP2:   LD A,21H        ;Initialize the ac-
                 0056                           ;cumulator
0106    DBC5     0057           IN A,(0C5H)     ;Input a byte of
                 0058                           ;data from port C5
0108    18FA     0059           JR LOOP2        ;Repeat until break
                 0060                           ;or reset
                 0061  ;
                 0062  ;
                 0063  ;
                 0064           NAME DECODE
010A    0E20     0065  DECODE:  LD C,20H        ;Load the device
                 0066                           ;code into regis-
                 0067                           ;ter C
010C    06C5     0068           LD B,0C5H       ;Load a nice look-
                 0069                           ;ing byte into reg-
                 0070                           ;ister B for sub-
                 0071                           ;sequent observa-
                 0072                           ;tion on the upper
                 0073                           ;half of the ad-
                 0074                           ;dress bus
010E    ED61     0075  LOOP3:   OUT (C),H       ;Output the content
                 0076                           ;of the H register
                 0077                           ;to port pointed to
                 0078                           ;by register C
0110    18FC     0079           JR LOOP3        ;Repeat output in-
                 0080                           ;struction until
                 0081                           ;break or reset
                 0082  ;
                 0083  ;
                 0084  ;
                 0085           NAME PULSR
0112    0E20     0086  PULSR:   LD C,20H        ;Load register C
```

```
                0087                            ;with the device
                0088                            ;code
0114  21000F    0089            LD HL,TABLE     ;Load register
                0090                            ;pair HL with the
                0091                            ;starting memory
                0092                            ;address
0117  0608      0093            LD B,08H        ;Load register B
                0094                            ;with the byte
                0095                            ;counter
0119  D3C0      0096            OUT (0C0H),A    ;Clear the decade
                0097                            ;counter
011B  EDB3      0098            OTIR            ;Output the byte
                0099                            ;string beginning
                0100                            ;at address HL of
                0101                            ;length (B) to port
                0102                            ;(C)
011D  76        0103            HALT            ;Halt the CPU
                0104    ;
                0105    ;       NAME MEM1
                0106    ;
011E  3EFF      0107    MEM1:   LD A,0FFH       ;Initialize the accu-
                0108                            ;mulator
0120  3C        0109    LOOP4:  INC A           ;Begin memory test for
                0110                            ;next value
0121  32007F    0111            LD (7F00H),A    ;Initialize location
                0112                            ;7F00 to contents of A
0124  01FF00    0113            LD BC,00FFH     ;BC = byte counter for
                0114                            ;LDIR instruction
0127  11017F    0115            LD DE,7F01H     ;DE = pointer to des-
                0116                            ;tination block
012A  21007F    0117            LD HL,7F00H     ;HL = pointer to source
                0118                            ;block
012D  EDB0      0119            LDIR            ;Load locations 7F00-
                0120                            ;7FFF with contents
                0121                            ;of register A
                0122    ;
012F  010001    0123    CHECK:  LD BC,0100H     ;Check that above load
                0124                            ;worked, BC = byte cnt
0132  21007F    0125            LD HL,7F00H     ;HL = pointer to loca-
                0126                            ;tion to be checked
0135  EDA1      0127    NXTLOC: CPI             ;Compare (HL) with
                0128                            ;contents of A
0137  200B      0129            JR NZ,ERROR     ;Mismatch indicates
                0130                            ;error
0139  E23E01    0131            JP PO,NEXXT     ;Parity flag = 0 in-
                0132                            ;dicates BC = 0000,
                0133                            ;Go to next test
                0134                            ;byte (INC A)
013C  18F7      0135            JR NXTLOC       ;Match and BC not =
                0136                            ;0000, go to next
                0137                            ;location
                0138    ;
013E  FEFF      0139    NEXXT:  CP 0FFH         ;See if A = FF.
0140  20DE      0140            JR NZ,LOOP4     ;If not, test
                0141                            ;next byte
```

```
0142  1820     0142              JR END              ;If so, test
               0143                                  ;is over
               0144    ;
0144  08       0145    ERROR:    EX AF,AF'           ;Display error byte
               0146                                  ;by using two rou-
               0147                                  ;tines from Nanocom-
               0148                                  ;puter operating
               0149                                  ;system
0145  3E70     0150              LD A,70H
0147  08       0151              EX AF,AF'
0148  3EE0     0152              LD A,0E0H
014A  32E50F   0153              LD (ADDH),A         ;Load 'E' in left-
               0154                                  ;most display digit
014D  2B       0155              DEC HL              ;HL = pointer to bad
               0156                                  ;location
014E  7D       0157              LD A,L
014F  32E20F   0158              LD (DATAL),A
0152  7C       0159              LD A,H
0153  32E30F   0160              LD (DATAH),A
0156  21B90F   0161              LD HL,LEDL
0159  11E50F   0162              LD DE,ADDH
015C  CD7CFA   0163              CALL CONVDI
015F  CD09F9   0164    ERRLP:    CALL DISPL
0162  18FB     0165              JR ERRLP
               0166    ;
0164  08       0167    END:      EX AF,AF'           ;Display F's if test OK
0165  3E00     0168              LD A,00H
0167  08       0169              EX AF,AF'
0168  3EFF     0170              LD A,0FFH
016A  32E50F   0171              LD (ADDH),A
016D  32E40F   0172              LD (ADDL),A
0170  32E30F   0173              LD (DATAH),A
0173  32E20F   0174              LD (DATAL),A
0176  21B90F   0175              LD HL,LEDL
0179  11E50F   0176              LD DE,ADDH
017C  CD7CFA   0177              CALL CONVDI
017F  CD09F9   0178    OK:       CALL DISPL
0182  18FB     0179              JR OK
               0180    ;
               0181    ;
               0182    ;
               0183              NAME XFER
0184  016600   0184    XFER:     LD BC,OK+5H—MEM1 ;Set-up for LDIR
               0185                                  ;OK+5H—MEM1 is the
               0186                                  ;number of bytes in pro-
                                                     ;gram MEM1
0187  11007F   0187              LD DE,7F00H         ;Destination is static RAM
               0188
018A  211E01   0189              LD HL,MEM1          ;Source block is
               0190                                  ;MEM1 program
018D  EDB0     0191              LDIR                ;Do it
018F  FF       0192              RST 38H             ;Return control
               0193                                  ;to the Nanocom-
               0194                                  ;puter operating
               0195                                  ;system
```

```
                0196    ;
                0197    ;
                0198    ;
                0199              NAME UCINP
0190   D311     0200    UCINP:    OUT (11H),A      ;Latch data from
                0201                               ;logic switches
0192   CD9A01   0202              CALL WAIT        ;Delay for awhile
0195   0E12     0203              LD C,12H         ;Set up C register
                0204                               ;with input device
                0205                               ;code
0197   ED40     0206              IN B,(C)         ;Input data from
                0207                               ;latch into B regis-
                0208                               ;ter by enabling
                0209                               ;the buffers
0199   FF       0210              RST 38H          ;Return control to
                0211                               ;the Nanocomputer
                0212                               ;operating system
                0213    ;
019A   210500   0214    WAIT:     LD HL,0005H      ;Delay loop
019D   11FFFF   0215    LOOP5:    LD DE,0FFFFH
01A0   1B       0216    LOOP6:    DEC DE
01A1   7A       0217              LD A,D
01A2   B3       0218              OR E
01A3   20FB     0219              JR NZ,LOOP6
01A5   2B       0220              DEC HL
01A6   7D       0221              LD A,L
01A7   B4       0222              OR H
01A8   20F3     0223              JR NZ,LOOP5
01AA   C9       0224              RET
                0225    ;
                0226    ;
                0227    ;
                0228              NAME UCINM
01AB   0E13     0229    UCINM:    LD C,13H         ;Set up 13 as the
                0230                               ;device code
01AD   ED40     0231    PCNTR:    IN B,(C)         ;Input pulse count
                0232                               ;to register B
01AF   ED41     0233              OUT (C),B        ;Output count to
                0234                               ;LEDs
01B1   CD9A01   0235              CALL WAIT        ;Delay before next
                0236                               ;count reading
01B4   18F7     0237              JR PCNTR         ;Repeat read/write/
                0238                               ;wait cycle
                0239    ;
                0240    ;
                0241    ;
                0242              NAME DDRIVE
01B6   010500   0243    DDRIVE:   LD BC,0005H      ;B contains
                0244                               ;data to be
                0245                               ;displayed
                0246                               ;C contains
                0247                               ;device code
                0248                               ;for output
                0249                               ;port (PIO
                0250                               ;#1 B, data)
```

```
01B9   3E00     0251            LD A,PSEL      ;A contains
                0252                           ;the display
                0253                           ;position
                0254                           ;selector
01BB   00       0255            NOP            ;Filler so this
                0256                           ;program will
                0257                           ;fit inside of
                0258                           ;next program
                0259                           ;without having
                0260                           ;to reload most
                0261                           ;of the bytes
01BC   ED79     0262            OUT (C),A      ;Output display
                0263                           ;address to
                0264                           ;the HCF4514 by
                0265                           ;toggling bit
                0266                           ;D0
01BE   3C       0267            INC A
01BF   ED79     0268            OUT (C),A
01C1   3D       0269            DEC A
01C2   ED79     0270            OUT (C),A
01C4   ED41     0271            OUT (C),B      ;Output data
01C6   76       0272            HALT
                0273   ;
                0274   ;
                0275   ;
                0276            NAME DISTST
01C7   010500   0277   DISTST:  LD BC,0005H    ;B contains data
                0278                           ;to be displayed
                0279                           ;C contains out-
                0280                           ;put device code
01CA   AF       0281   DATALP:  XOR A          ;A contains the
                0282                           ;position to be
                0283                           ;displayed
01CB   160A     0284            LD D,0AH       ;D is the display
                0285                           ;position counter
01CD   ED79     0286   OUTPUT:  OUT (C),A      ;Output display ad-
                0287                           ;dress to HCF4514
                0288                           ;by toggling bit D0
01CF   3C       0289            INC A
01D0   ED79     0290            OUT (C),A
01D2   3D       0291            DEC A
01D3   ED79     0292            OUT (C),A
01D5   ED41     0293            OUT (C),B      ;Output data
                0294
01D7   3C       0295            INC A          ;Increment position
                0296                           ;pointer to point to
                0297                           ;next display posi-
                0298                           ;tion
01D8   3C       0299            INC A
01D9   CDE301   0300            CALL DELAY     ;Pause so display is
                0301                           ;constant for a short
                0302                           ;period
01DC   15       0303            DEC D          ;Decrement position
                0304                           ;counter
01DD   20EE     0305            JR NZ,OUTPUT   ;If D is not zero, then
```

```
                0306                            ;go back to output byte
                0307                            ;to next display posi-
                0308                            ;tion.
01DF  04        0309            INC B           ;If all display posi-
                0310                            ;tions have been tested,
                0311                            ;update the output
                0312                            ;data
01E0  04        0313            INC B
01E1  18E7      0314            JR DATALP       ;Start again with new
                0315                            ;data byte
                0316    ;
01E3  D5        0317    DELAY:  PUSH DE         ;Save DE
01E4  16F0      0318            LD D,0F0H       ;Timing byte
01E6  CDF2F9    0319    DREGL:  CALL BAUD       ;BAUD is a routine
                0320                            ;in the operating
                0321                            ;system that delays
                0322                            ;exactly one sampling
                0323                            ;period. The
                0324                            ;length of the pe-
                0325                            ;riod is set via a
                0326                            ;timing byte stored
                0327                            ;in memory. In sub-
                0328                            ;routine DELAY, the
                0329                            ;delay will be 16
                0330                            ;(base 10) sampling
                0331                            ;periods
01E9  15        0332            DEC D
01EA  20FA      0333            JR NZ,DREGL
01EC  D1        0334            POP DE          ;Restore DE
01ED  C9        0335            RET
                0336    ;
                0337    ;
                0338    ;
                0339            NAME KBTST
01EE  CD9DF9    0340    KBTST:  CALL CHECKB     ;Check for pressed
                0341                            ;key
01F1  28FB      0342            JR Z,KBTST      ;Z-flag = 1 implies
                0343                            ;that no key is
                0344                            ;pressed
01F3  CDDBF8    0345    GETNO:  CALL KBSCAN     ;Z-flag = 0 implies
                0346                            ;that one or more
                0347                            ;keys are pressed.
                0348                            ;See if just one,
                0349                            ;and which one.
01F6  38F6      0350            JR C,KBTST      ;C-flag = 1 implies
                0351                            ;that more than one
                0352                            ;key was pressed
01F8  32E20F    0353            LD (DATAL),A    ;C-flag = 0 implies
                0354                            ;that one key was
                0355                            ;pressed and its
                0356                            ;number is in reg-
                0357                            ;ister A. Display
                0358                            ;hex key number in
                0359                            ;data display posi-
                0360                            ;tions
```

```
01FB   08        0361              EX AF,AF'         ;Set up for call
                 0362                                ;to CONDVI
01FC   3EFC      0363              LD A,0FCH         ;Just display data
                 0364                                ;digits
01FE   08        0365              EX AF,AF'
01FF   11E50F    0366              LD DE,ADDH
0202   21B90F    0367              LD HL,ADD7-1
0205   CD7CFA    0368              CALL CONVDI       ;Translate key no
                 0369                                ;for display
0208   CD09F9    0370    DSPLAY:   CALL DISPL        ;Display the key
                 0371                                ;number
020B   CD9DF9    0372              CALL CHECKB       ;Check for pressed
                 0373                                ;key
020E   28F8      0374              JR Z,DSPLAY       ;Keep displaying
                 0375                                ;if no key pressed
0210   18E1      0376              JR GETNO          ;Get key number
                 0377                                ;if key is pressed
                 0378    ;
                 0379    ;
                 0380    ;
                 0381              NAME OUTSIM
0212   3E0F      0382    OUTSIM:   LD A,0FH          ;Program the PIO #2 to
                 0383                                ;Mode 0
0214   D30A      0384              OUT (0AH),A
0216   3E43      0385              LD A,43H          ;Output the byte 43H
                 0386                                ;to PC0-7 lines
0218   D308      0387              OUT (08H),A
021A   76        0388              HALT
                 0389    ;
                 0390    ;
                 0391    ;
                 0392              NAME INIT0
021B   3EC3      0393    INIT0:    LD A,0C3H         ;first byte is jump
021D   323800    0394              LD (0038H),A      ;load into RST location
0220   FD216E02  0395              LD IY,SERV1       ;address of service
0224   FD223900  0396              LD (0039H),IY     ;routine #1
0228   ED46      0397              IM0               ;Interrupt Mode 0
022A   08        0398              EX AF,AF'         ;set format for blanks
022B   3E40      0399              LD A,40H          ;for CONVDI
022D   08        0400              EX AF,AF'
022E   C3C302    0401              JP MAIN           ;Jump to routine MAIN
                 0402    ;
                 0403    ;
                 0404    ;
                 0405              NAME INIT1
0231   3EC3      0406    INIT1:    LD A,0C3H         ;first byte is jump
0233   323800    0407              LD (0038H),A
0236   FD216E02  0408              LD IY,SERV1       ;address of service
023A   FD223900  0409              LD (0039H),IY     ;routine #1
023E   ED56      0410              IM1               ;Interrupt mode 1
0240   08        0411              EX AF,AF'         ;set format for blanks
0241   3E40      0412              LD A,40H          ;for CONVDI
0243   08        0413              EX AF,AF'
0244   C3C302    0414              JP MAIN           ;Jump to routine MAIN
                 0415    ;
```

```
                    0416   ;
                    0417   ;
                    0418              NAME INIT2
0247   ED5E         0419   INIT2:     IM2               ;Interrupt mode 2
0249   21000F       0420              LD HL,TABLE       ;address of vector table
024C   7C           0421              LD A,H            ;high byte of address
024D   ED47         0422              LD I,A            ;set Interrupt register
024F   FD216E02     0423              LD IY,SERV1       ;first service routine
0253   FD22000F     0424              LD (TABLE),IY     ;set in vector table
0257   FD21F502     0425              LD IY,SERV2       ;second service routine
025B   FD22020F     0426              LD (TABLE+2),IY      ;set in vector table
025F   FD216B03     0427              LD IY,SERV3       ;third service routine
0263   FD22040F     0428              LD (TABLE+4),IY      ;set in vector table
0267   08           0429              EX AF,AF'         ;set format for CONVDI
0268   3E40         0430              LD A,40H
026A   08           0431              EX AF,AF'
026B   C3C302       0432              JP MAIN           ;Jump to routine MAIN
                    0433   ;
                    0434   ;
                    0435   ;
                    0436              NAME SERV1
026E   C5           0437   SERV1:     PUSH BC           ;save CPU registers
026F   D5           0438              PUSH DE
0270   E5           0439              PUSH HL
0271   F5           0440              PUSH AF
0272   DDE5         0441              PUSH IX
0274   FDE5         0442              PUSH IY
0276   DD23         0443   DS1:       INC IX            ;update data stack pointer
0278   DD23         0444              INC IX
027A   DD23         0445              INC IX
027C   00           0446              NOP               ;no operation
027D   DD3600FF     0447              LD (IX+00H),0FFH   ;set DLOOP1 time
0281   DD36010A     0448              LD (IX+01H),00AH   ;set CLOOP1 time
0285   DD360202     0449   CLOOP1:    LD (IX+02H),02H    ;set DLOOP1 time
0289   21E50F       0450              LD HL,ADDH        ;point to display buffer
028C   ED57         0451              LD A,I            ;find value of IFF2
028E   EA9502       0452              JP PE,HIGH1
0291   3600         0453   LOW1:      LD (HL),00H       ;value = 0
0293   1802         0454              JR NEXT1
0295   3610         0455   HIGH1:     LD (HL),10H       ;value = 1
0297   2B           0456   NEXT1:     DEC HL            ;move buffer pointer
0298   34           0457              INC (HL)          ;increment ADDL
0299   ED73E20F     0458              LD (DATAL),SP     ;copy SP to buffer
029D   21B90F       0459              LD HL,LEDL        ;set for CONVDI
02A0   11E50F       0460              LD DE,ADDH        ;set for CONVDI
02A3   CD7CFA       0461              CALL CONVDI
02A6   CD09F9       0462   DLOOP1:    CALL DISPL
02A9   DD3500       0463              DEC (IX+00)       ;timer for display
02AC   20F8         0464              JR NZ,DLOOP1
02AE   DD3502       0465              DEC (IX+02)       ;timer for display
02B1   20F3         0466              JR NZ,DLOOP1
02B3   DD3501       0467              DEC (IX+01)       ;timer for service routine
02B6   20CD         0468              JR NZ,CLOOP1
02B8   FDE1         0469              POP IY            ;restore CPU registers
02BA   DDE1         0470              POP IX
```

```
02BC   F1         0471           POP AF
02BD   E1         0472           POP HL
02BE   D1         0473           POP DE
02BF   C1         0474           POP BC
02C0   FB         0475           EI                  ;enable interrupts
02C1   ED4D       0476           RETI                ;return from interrupt
                  0477   ;
                  0478   ;
                  0479   ;
                  0480           NAME MAIN
02C3   FB         0481   MAIN:   EI                  ;enable interrupts
02C4   DD21000C   0482           LD IX,DSTACK        ;bottom of data stack
02C8   DD3600FF   0483           LD (IX+00H),0FFH    ;timer for display
02CC   21E50F     0484           LD HL,ADDH          ;set pointer to buffer
02CF   ED57       0485           LD A,I              ;find value of IFF2
02D1   EAD802     0486           JP PE,HIGH
02D4   3600       0487   LOW:    LD (HL),00H         ;value = 0
02D6   1802       0488           JR NEXT
02D8   3610       0489   HIGH:   LD (HL),10H         ;value = 1
02DA   2B         0490   NEXT:   DEC HL              ;move buffer pointer
02DB   35         0491           DEC (HL)            ;decrement COUNT
02DC   ED73E20F   0492           LD (DATAL),SP       ;copy SP to buffer
02E0   21B90F     0493           LD HL,LEDL          ;set up for CONVDI
02E3   11E50F     0494           LD DE,ADDH          ;set up for CONVDI
02E6   00         0495   DISAB:  NOP                 ;no operation
02E7   CD7CFA     0496           CALL CONVDI
02EA   CD09F9     0497   DLOOP:  CALL DISPL
02ED   DD3500     0498           DEC (IX+00H)        ;timer for display
02F0   20F8       0499           JR NZ,DLOOP
02F2   C3C302     0500           JP MAIN             ;jump back to beginning
                  0501   ;
                  0502   ;
                  0503   ;
                  0504           NAME SERV2
02F5   76         0505   SERV2:  HALT                ;Halt the microcomputer
                  0506   ;
                  0507   ;
                  0508   ;
                  0509           NAME INIT1N
02F6   3EC3       0510   INIT1N: LD A,0C3H           ;first byte is jump
02F8   326600     0511           LD (0066H),A        ;non-maskable interrupt
02FB   FD211903   0512           LD IY,SERVN         ;address of service for
02FF   FD226700   0513           LD (0067H),IY       ;non-maskable interrupt
0303   ED56       0514           IM1                 ;Interrupt mode 1
0305   3EC3       0515           LD A,0C3H           ;first byte is jump
0307   323800     0516           LD (0038H),A
030A   FD216E02   0517           LD IY,SERV1         ;address of service
030E   FD223900   0518           LD (0039H),IY       ;routine #1
0312   08         0519           EX AF,AF'           ;set format for blanks
0313   3E40       0520           LD A,40H            ;for CONVDI
0315   08         0521           EX AF,AF'
0316   C3C302     0522           JP MAIN             ;Jump to routine MAIN
                  0523   ;
                  0524   ;
                  0525   ;
```

```
                 0526                NAME SERVN
0319   C5        0527   SERVN:       PUSH BC           ;save CPU registers
031A   D5        0528                PUSH DE
031B   E5        0529                PUSH HL
031C   F5        0530                PUSH AF
031D   DDE5      0531                PUSH IX
031F   FDE5      0532                PUSH IY
0321   DD23      0533   DSN:         INC IX            ;update data stack point
0323   DD23      0534                INC IX
0325   DD23      0535                INC IX
0327   00        0536                NOP               ;no operation
0328   DD3600FF  0537                LD (IX+00H),0FFH  ;set DLOOPN time
032C   DD36010A  0538                LD (IX+01H),00AH  ;set CLOOPN time
0330   DD360202  0539   CLOOPN:      LD (IX+02H),02H   ;set DLOOPN time
0334   21E50F    0540                LD HL,ADDH        ;point to display buffer
0337   ED57      0541                LD A,I            ;find value of IFF2
0339   EA4003    0542                JP PE,HIGHN
033C   3600      0543   LOWN:        LD (HL),00H       ;value = 0
033E   1802      0544                JR NEXTN
0340   3610      0545   HIGHN:       LD (HL),10H       ;value = 1
0342   ED73E20F  0546   NEXTN:       LD (DATAL),SP     ;copy SP to buffer
0346   21B90F    0547                LD HL,LEDL        ;set for CONVDI
0349   11E50F    0548                LD DE,ADDH        ;set for CONVDI
034C   CD7CFA    0549                CALL CONVDI
034F   CD09F9    0550   DLOOPN:      CALL DISPL
0352   DD3500    0551                DEC (IX+00)       ;timer for display
0355   20F8      0552                JR NZ,DLOOPN
0357   DD3502    0553                DEC (IX+02)       ;timer for display
035A   20F3      0554                JR NZ,DLOOPN
035C   DD3501    0555                DEC (IX+01)       ;timer for service routine
035F   20CF      0556                JR NZ,CLOOPN
0361   FDE1      0557                POP IY            ;restore CPU registers
0363   DDE1      0558                POP IX
0365   F1        0559                POP AF
0366   E1        0560                POP HL
0367   D1        0561                POP DE
0368   C1        0562                POP BC
0369   ED45      0563                RETN              ;return from non-maskable
                 0564                                  ;interrupt
                 0565   ;
                 0566   ;
                 0567   ;
                 0568                NAME SERV3
036B   C5        0569   SERV3:       PUSH BC           ;save CPU registers
036C   D5        0570                PUSH DE
036D   E5        0571                PUSH HL
036E   F5        0572                PUSH AF
036F   DDE5      0573                PUSH IX
0371   FDE5      0574                PUSH IY
0373   DD23      0575   DS3:         INC IX            ;update data stack pointer
0375   DD23      0576                INC IX
0377   DD23      0577                INC IX
0379   00        0578                NOP               ;no operation
037A   DD3600FF  0579                LD (IX+00H),0FFH  ;set DLOOP3 time
037E   DD36010A  0580                LD (IX+01H),00AH  ;set CLOOP3 time
```

```
0382  DD360202  0581  CLOOP3:  LD (IX+02H),02H   ;set DLOOP3 time
0386  21E50F    0582           LD HL,ADDH        ;point to display buffer
0389  ED57      0583           LD A,I            ;find value of IFF2
038B  EA9203    0584           JP PE,HIGH3
038E  3600      0585  LOW3:    LD (HL),00H       ;value = 0
0390  1802      0586           JR NEXT3
0392  3610      0587  HIGH3:   LD (HL),10H       ;value = 1
0394  2B        0588  NEXT3:   DEC HL            ;move buffer pointer
0395  34        0589           INC (HL)          ;increment ADDL
0396  34        0590           INC (HL)          ;increment ADDL
0397  ED73E20F  0591           LD (DATAL),SP     ;copy SP to buffer
039B  21B90F    0592           LD HL,LEDL        ;set for CONVDI
039E  11E50F    0593           LD DE,ADDH        ;set for CONVDI
03A1  CD7CFA    0594           CALL CONVDI
03A4  CD09F9    0595  DLOOP3:  CALL DISPL
03A7  DD3500    0596           DEC (IX+00)       ;timer for display
03AA  20F8      0597           JR NZ,DLOOP3
03AC  DD3502    0598           DEC (IX+02)       ;timer for display
03AF  20F3      0599           JR NZ,DLOOP3
03B1  DD3501    0600           DEC (IX+01)       ;timer for service routine
03B4  20CC      0601           JR NZ,CLOOP3
03B6  FDE1      0602           POP IY            ;restore CPU registers
03B8  DDE1      0603           POP IX
03BA  F1        0604           POP AF
03BB  E1        0605           POP HL
03BC  D1        0606           POP DE
03BD  C1        0607           POP BC
03BE  FB        0608           EI                ;enable interrupts
03BF  ED4D      0609           RETI              ;return from interrupt
                0610  ;
                0611  ;
                0612  ;
                0613           NAME INITOC
03C1  ED5E      0614  INITOC:  IM2               ;set Z80 interrupt mode
03C3  21000F    0615           LD HL,TABLE       ;address of vector table
03C6  7C        0616           LD A,H            ;high byte of address
03C7  ED47      0617           LD I,A            ;set interrupt register
03C9  FD21E803  0618           LD IY,SERVOC      ;PIO output service routine
03CD  FD22060F  0619           LD (TABLE+06H),IY ;set in vector table
03D1  3E06      0620           LD A,06H          ;Load interrupt vector
03D3  D30A      0621           OUT (0AH),A       ;for port C
03D5  08        0622           EX AF,AF'         ;set format for CONVDI
03D6  3E40      0623           LD A,40H
03D8  08        0624           EX AF,AF'
03D9  3E0F      0625           LD A,0FH          ;Set PIO mode
03DB  D30A      0626           OUT (0AH),A
03DD  3E87      0627  ENPIO:   LD A,87H          ;Enable PIO interrupts
03DF  D30A      0628           OUT (0AH),A
03E1  3EFF      0629           LD A,0FFH         ;initialize CRDY signal
03E3  D308      0630  THROW:   OUT (08H),A
03E5  C3C302    0631           JP MAIN           ;jump to routine MAIN
                0632  ;
                0633  ;
                0634  ;
                0635           NAME SERVOC
```

```
03E8   E5         0636  SERVOC:  PUSH HL          ;save CPU register status
03E9   F5         0637           PUSH AF
03EA   3AE40F     0638           LD A,(ADDL)      ;move buffer value to A
03ED   D308       0639           OUT (08H),A      ;output buffer value
03EF   F1         0640           POP AF           ;restore CPU register status
03F0   E1         0641           POP HL
03F1   FB         0642           EI               ;enable interrupts
03F2   ED4D       0643           RETI             ;return from interrupt
                  0644  ;
                  0645  ;
                  0646  ;
                  0647           NAME INITID
03F4   ED5E       0648  INITID:  IM2              ;Interrupt mode 2
03F6   21000F     0649           LD HL,TABLE      ;address of vector table
03F9   7C         0650           LD A,H           ;high byte of address
03FA   ED47       0651           LD I,A           ;set interrupt register
03FC   FD211F04   0652           LD IY,SERVID     ;input service routine
0400   FD22080F   0653           LD (TABLE+08H),IY ;set in vector table
0404   3E08       0654           LD A,08H         ;Load interrupt vector
0406   D30B       0655           OUT (0BH),A
0408   08         0656           EX AF,AF'        ;set format for CONVDI
0409   3E40       0657           LD A,40H
040B   08         0658           EX AF,AF'
040C   3E4F       0659           LD A,4FH         ;Set PIO mode
040E   D30B       0660           OUT (0BH),A
0410   3E87       0661           LD A,87H         ;enable PIO interrupt
0412   D30B       0662           OUT (0BH),A
0414   DB09       0663           IN A,(09H)       ;initialize DRDY
0416   C3C302     0664           JP MAIN
                  0665  ;
                  0666  ;
                  0667  ;
                  0668           NAME SERVIC
0419   C5         0669  SERVIC:  PUSH BC          ;save BC
041A   0E08       0670           LD C,08H         ;PORT C interrupt
041C   C33104     0671           JP SERVI
                  0672  ;
                  0673  ;
                  0674  ;
                  0675           NAME SERVID
041F   C5         0676  SERVID:  PUSH BC
0420   0E09       0677           LD C,09H         ;PORT D interrupt
0422   C33104     0678           JP SERVI
                  0679  ;
                  0680  ;
                  0681  ;
                  0682           NAME SERVIE
0425   C5         0683  SERVIE:  PUSH BC
0426   0E0C       0684           LD C,0CH         ;PORT E interrupt
0428   C33104     0685           JP SERVI
                  0686  ;
                  0687  ;
                  0688  ;
                  0689           NAME SERVIF
042B   C5         0690  SERVIF:  PUSH BC
```

```
042C  0E0D      0691              LD C,0DH          ;PORT F Interrupt
042E  C33104    0692              JP SERVI
                0693    ;
                0694    ;
                0695    ;
                0696              NAME SERVI
0431  00        0697    SERVI:    NOP               ;previously saved BC
0432  D5        0698              PUSH DE
0433  E5        0699              PUSH HL
0434  F5        0700              PUSH AF
0435  DDE5      0701              PUSH IX
0437  FDE5      0702              PUSH IY
0439  FD2AE40F  0703              LD IY,(ADDL)      ;save state of (ADDL)
043D  FDE5      0704              PUSH IY
043F  ED78      0705              IN A,(C)
0441  32E40F    0706              LD (ADDL),A       ;put byte in ADDL
0444  DD23      0707    DSG:      INC IX            ;update data stack pointer
0446  DD23      0708              INC IX
0448  DD23      0709              INC IX
044A  00        0710    ENABG:    NOP               ;no operation
044B  DD3600FF  0711              LD (IX+00H),0FFH  ;set DLOOPG time
044F  DD36010A  0712              LD (IX+01H),00AH  ;set CLOOPG time
0453  DD360202  0713    CLOOPG:   LD (IX+02H),02H   ;set DLOOPG time
0457  21E50F    0714              LD HL,ADDH        ;point to display buffer
045A  ED57      0715              LD A,I            ;find value of IFF2
045C  EA6304    0716              JP PE,HIGHG
045F  3600      0717    LOWG:     LD (HL),00H       ;value = 0
0461  1802      0718              JR NEXTG
0463  3610      0719    HIGHG:    LD (HL),10H       ;value = 1
0465  ED73E20F  0720    NEXTG:    LD (DATAL),SP     ;copy SP to buffer
0469  21B90F    0721              LD HL,LEDL        ;set for CONVDI
046C  11E50F    0722              LD DE,ADDH        ;set for CONVDI
046F  CD7CFA    0723              CALL CONVDI
0472  CD09F9    0724    DLOOPG:   CALL DISPL
0475  DD3500    0725              DEC (IX+00)       ;timer for display
0478  20F8      0726              JR NZ,DLOOPG
047A  DD3502    0727              DEC (IX+02)       ;timer for display
047D  20F3      0728              JR NZ,DLOOPG
047F  DD3501    0729              DEC (IX+01)       ;timer for service routine
0482  20CF      0730              JR NZ,CLOOPG
0484  FDE1      0731              POP IY            ;restore contents of ADDL
0486  FD22E40F  0732              LD (ADDL),IY
048A  FDE1      0733              POP IY            ;restore CPU registers
048C  DDE1      0734              POP IX
048E  F1        0735              POP AF
048F  E1        0736              POP HL
0490  D1        0737              POP DE
0491  C1        0738              POP BC
0492  FB        0739              EI                ;enable interrupts
0493  ED4D      0740              RETI              ;return from interrupts
                0741    ;
                0742    ;
                0743    ;
                0744              NAME INITPB
0495  ED5E      0745    INITPB:   IM2               ;Z80 interrupt mode 2
```

```
0497  21000F    0746            LD HL,TABLE       ;address of vector table
049A  7C        0747            LD A,H            ;high byte of address
049B  ED47      0748            LD I,A            ;set interrupt register
049D  FD21E803  0749            LD IY,SERVOC      ;output service routine
04A1  FF22060F  0750            LD (TABLE+06H),IY ;set in vector table
04A5  FD211904  0751            LD IY,SERVIC      ;input service routine
04A9  FD220A0F  0752            LD (TABLE+0AH),IY ;set in vector table
04AD  3E06      0753            LD A,06H          ;load interrupt vector
04AF  D30A      0754            OUT (0AH),A       ;for port C
04B1  3E0A      0755            LD A,0AH          ;load interrupt vector
04B3  D30B      0756            OUT (0BH),A       ;for port D
04B5  08        0757            EX AF,AF'         ;set format for CONVDI
04B6  3E40      0758            LD A,40H          ;
04B8  08        0759            EX AF,AF'         ;
04B9  3E8F      0760            LD A,8FH          ;set PIO mode 2
04BB  D30A      0761            OUT (0AH),A       ;port C
04BD  3ECF      0762            LD A,0CFH         ;set PIO mode 3
04BF  D30B      0763            OUT (0BH),A       ;port D
04C1  3EFF      0764            LD A,0FFH         ;set mask byte Port D re-
04C3  D30B      0765            OUT (0BH),A       ;quired to follow set PIO
04C5  3E87      0766            LD A,87H          ;mode 3
04C7  D30A      0767            OUT (0AH),A       ;enable PIO interrupts
                                                  ;port C
04C9  D30B      0768            OUT (0BH),A       ;port D
04CB  3EFF      0769            LD A,0FFH         ;initialize CRDY
04CD  D308      0770            OUT (08H),A
04CF  DB08      0771            IN A,(08H)        ;initialize DRDY
04D1  C3C302    0772            JP MAIN           ;jump to routine MAIN
                0773  ;
                0774  ;
                0775  ;
                0776            NAME INITPM
04D4  ED5E      0777  INITPM:   IM2               ;Z80 interrupt mode 2
04D6  21000F    0778            LD HL,TABLE       ;address of vector table
04D9  7C        0779            LD A,H            ;high byte of address
04DA  ED47      0780            LD I,A            ;set interrupt register
04DC  FD210505  0781            LD IY,SERVM       ;address of service routine
04E0  FD220C0F  0782            LD (TABLE+0CH),IY ;set in vector table
04E4  3E0C      0783            LD A,0CH          ;set interrupt vector for
04E6  D30B      0784            OUT (0BH),A       ;port D
04E8  08        0785            EX AF,AF'         ;set format for CONVDI
04E9  3E40      0786            LD A,40H
04EB  08        0787            EX AF,AF'
04EC  3ECF      0788            LD A,0CFH         ;set mode 3 for port D
04EE  D30B      0789            OUT (0BH),A
04F0  3E0F      0790            LD A,0FH          ;define input lines for
04F2  D30B      0791            OUT (0BH),A       ;port D
04F4  3E97      0792  CWORD:    LD A,97H          ;set interrupt control word
04F6  D30B      0793            OUT (0BH),A
04F8  3EFC      0794            LD A,0FCH         ;monitor PB0,PB1
04FA  D30B      0795            OUT (0BH),A
04FC  0E09      0796            LD C,(09H)        ;initialize lamp monitors
04FE  3E00      0797            LD A,00H          ;to off position
0500  ED79      0798            OUT (C),A
0502  C3C302    0799            JP MAIN
```

```
                    0800  ;
                    0801  ;
                    0802  ;
                    0803             NAME SERVM
0505  C5            0804  SERVM:     PUSH BC             ;save CPU registers
0506  D5            0805             PUSH DE
0507  E5            0806             PUSH HL
0508  F5            0807             PUSH AF
0509  DDE5          0808             PUSH IX
050B  FDE5          0809             PUSH IY
050D  FD2AE40F      0810             LD IY,(ADDL)        ;save state of (ADDL)
0511  FDE5          0811             PUSH IY
0513  0E09          0812             LD C,09H            ;input from PIO port C
0515  ED78          0813             IN A,(C)
0517  E60F          0814             AND 0FH             ;clear high order nibble
0519  32E40F        0815             LD (ADDL),A         ;put byte in ADDL
051C  17            0816             RLA                 ;transpose high order
051D  17            0817             RLA                 ;nibble with low order
051E  17            0818             RLA                 ;nibble
051F  17            0819             RLA
0520  ED79          0820             OUT (C),A           ;output to lamp monitors
0522  DD23          0821  DSM:       INC IX              ;update data stack pointer
0524  DD23          0822             INC IX
0526  DD23          0823             INC IX
0528  00            0824             NOP                 ;no operation
0529  DD3600FF      0825             LD (IX+00H),0FFH    ;set inner DLOOPM time
052D  DD36010A      0826             LD (IX+01H),00AH    ;set CLOOPM time
0531  DD360202      0827  CLOOPM:    LD (IX+02H),02H     ;set outer DLOOPM time
0535  21E50F        0828             LD HL,ADDH          ;point to display buffer
0538  ED57          0829             LD A,I              ;find value of IFF2
053A  EA4105        0830             JP PE,HIGHM
053D  3600          0831  LOWM:      LD (HL),00H         ;value = 0
053F  1802          0832             JR NEXTM
0541  3610          0833  HIGHM:     LD (HL),10H         ;value = 1
0543  ED73E20F      0834  NEXTM:     LD (DATAL),SP       ;copy SP to buffer
0547  21B90F        0835             LD HL,LEDL          ;set for CONVDI
054A  11E50F        0836             LD DE,ADDH          ;set for CONVDI
054D  CD7CFA        0837             CALL CONVDI
0550  CD09F9        0838  DLOOPM:    CALL DISPL
0553  DD3500        0839             DEC (IX+00)         ;timer for display
0556  20F8          0840             JR NZ,DLOOPM
0558  DD3502        0841             DEC (IX+02)         ;timer for display
055B  20F3          0842             JR NZ,DLOOPM
055D  DD3501        0843             DEC (IX+01)         ;timer for service routine
0560  20CF          0844             JR NZ,CLOOPM
0562  FDE1          0845             POP IY              ;restore contents of ADDL
0564  FD22E40F      0846             LD (ADDL),IY
0568  FDE1          0847             POP IY              ;restore CPU registers
056A  DDE1          0848             POP IX
056C  F1            0849             POP AF
056D  E1            0850             POP HL
056E  D1            0851             POP DE
056F  C1            0852             POP BC
0570  FB            0853             EI                  ;enable interrupts
0571  ED4D          0854             RETI                ;return from interrupt
```

```
                    0855   ;
                    0856   ;
                    0857   ;
                    0858               NAME INITPP
0573  ED5E          0859   INITPP:     IM2                  ;Z80 mode 2 interrupts
0575  21000F        0860               LD HL,TABLE          ;address of vector table
0578  7C            0861               LD A,H               ;high byte of address
0579  ED47          0862               LD I,A               ;set Interrupt vector
057B  FD211904      0863               LD IY,SERVIC         ;service for port C
057F  FD220A0F      0864               LD (TABLE+0AH),IY ;set in table
0583  FD211F04      0865               LD IY,SERVID         ;port D
0587  FD22080F      0866               LD (TABLE+08H),IY  ;set in table
058B  3E0A          0867               LD A,0AH             ;set interrupt vector for C
058D  D30A          0868               OUT (0AH),A
058F  3E08          0869               LD A,08H             ;set interrupt vector for D
0591  D30B          0870               OUT (0BH),A
0593  08            0871               EX AF,AF'            ;set format for CONVDI
0594  3E40          0872               LD A,40H
0596  08            0873               EX AF,AF'
0597  3E4F          0874               LD A,4FH             ;mode 1 for C and D
0599  D30A          0875               OUT (0AH),A
059B  D30B          0876               OUT (0BH),A
059D  3E87          0877               LD A,87H             ;enable C and D
059F  D30A          0878               OUT (0AH),A
05A1  D30B          0879               OUT (0BH),A
05A3  DB08          0880               IN A,(08H)           ;initialize CRDY
05A5  DB09          0881               IN A,(09H)           ;and DRDY
05A7  C3C302        0882               JP MAIN
                    0883   ;
                    0884   ;
                    0885   ;
                    0886               NAME INITDC
05AA  ED5E          0887   INITDC:     IM2                  ;Z80 interrupt mode 2
05AC  21000F        0888               LD HL,TABLE          ;address of vector table
05AF  7C            0889               LD A,H               ;high byte of address
05B0  ED47          0890               LD I,A               ;set Interrupt vector
05B2  FD212504      0891               LD IY,SERVIE         ;service routine port E input
05B6  FD220E0F      0892               LD (TABLE+0EH),IY  ;set in table
05BA  FD212B04      0893               LD IY,SERVIF         ;service routine port F input
05BE  FD22100F      0894               LD (TABLE+10H),IY  ;set in table
05C2  3E0E          0895               LD A,0EH             ;load interrupt vector E
05C4  D30E          0896               OUT (0EH),A
05C6  3E10          0897               LD A,10H             ;load interrupt vector F
05C8  D30F          0898               OUT (0FH),A
05CA  08            0899               EX AF,AF'            ;set format for CONVDI
05CB  3E40          0900               LD A,40H
05CD  08            0901               EX AF,AF'
05CE  3E4F          0902               LD A,4FH             ;set PIO mode 1
05D0  D30E          0903               OUT (0EH),A          ;port E
05D2  D30F          0904               OUT (0FH),A          ;port F
05D4  3E87          0905               LD A,87H             ;enable PIO
05D6  D30E          0906               OUT (0EH),A          ;port E
05D8  D30F          0907               OUT (0FH),A          ;port F
05DA  DB0C          0908               IN A, (0CH)          ;initialize ERDY
05DC  DB0D          0909               IN A, (0DH)          ;initialize FRDY
```

```
05DE  C37305    0910              JP INITPP
                0911    ;
                0912    ;
                0913    ;
                0914              NAME SEROCX
05E1  C5        0915    SEROCX:   PUSH BC             ;save CPU registers
05E2  D5        0916              PUSH DE
05E3  E5        0917              PUSH HL
05E4  F5        0918              PUSH AF
05E5  DDE5      0919              PUSH IX
05E7  FDE5      0920              PUSH IY
05E9  FD2AE40F  0921              LD IY,(ADDL)        ;save state of (ADDL)
05ED  FDE5      0922              PUSH IY
05EF  DD23      0923    DSX:      INC IX              ;update data stack pointer
05F1  DD23      0924              INC IX
05F3  DD23      0925              INC IX
05F5  00        0926              NOP                 ;no operation
05F6  DD3600FF  0927              LD (IX+00H),0FFH    ;set DLOOPX time
05FA  DD36010A  0928              LD (IX+01H),00AH    ;set CLOOPX time
05FE  DD360201  0929    CLOOPX:   LD (IX+02H),01H     ;set DLOOPX time
0602  21E50F    0930              LD HL,ADDH          ;point to display buffer
0605  ED57      0931              LD A,I              ;find value of IFF2
0607  EA0E06    0932              JP PE,HIGHX
060A  3600      0933    LOWX:     LD (HL),00H         ;value = 0
060C  1802      0934              JR NEXTX
060E  3610      0935    HIGHX:    LD (HL),10H         value = 1
0610  2B        0936    NEXTX:    DEC HL              ;move buffer pointer
0611  34        0937              INC (HL)            ;increment ADDL
0612  ED73E20F  0938              LD (DATAL),SP       ;copy SP to buffer
0616  21B90F    0939              LD HL,LEDL          ;set for CONVDI
0619  11E50F    0940              LD DE,ADDH          ;set for CONVDI
061C  CD7CFA    0941              CALL CONVDI
061F  CD09F9    0942    DLOOPX:   CALL DISPL
0622  DD3500    0943              DEC (IX+00)         ;timer for display
0625  20F8      0944              JR NZ,DLOOPX
0627  DD3502    0945              DEC (IX+02)         ;timer for display
062A  20F3      0946              JR NZ,DLOOPX
062C  DD3501    0947              DEC (IX+01)         ;timer for service routine
062F  20CD      0948              JR NZ,CLOOPX
0631  FDE1      0949              POP IY              ;restore CPU registers
0633  FD22E40F  0950              LD (ADDL),IY        ;restore state of (ADDL)
0637  3AE40F    0951    OUTX:     LD A,(ADDL)         ;output the byte that was
063A  D308      0952              OUT (08H),A         ;in ADDL when interrupted
063C  FDE1      0953              POP IY              ;restore CPU registers
063E  DDE1      0954              POP IX
0640  F1        0955              POP AF
0641  E1        0956              POP HL
0642  D1        0957              POP DE
0643  C1        0958              POP BC
0644  FB        0959              EI                  ;enable interrupts
0645  ED4D      0960              RETI                ;return from interrupt
                0961    ;
                0962    ;
                0963    ;
                0964              NAME CHPTST
```

```
0647    3E03       0965   CHPTST:  LD A,03H         ;Reset PIO interrupt
                   0966                             ;enable FLIP-FLOP
0649    D30A       0967            OUT (0AH),A
064B    D30B       0968            OUT (0BH),A
064D    2A0300     0969            LD HL,(MASKW)    ;SET mask for circuit
0650    010AFF     0970            LD BC,0FF0AH
0653    ED41       0971            OUT (C),B        ;PIO Port A to mode 3
0655    ED69       0972            OUT (C),L        ;I/O mask for Port A
0657    0C         0973            INC C            ;Change to Port B
0658    ED41       0974            OUT (C),B        ;PIO Port B to mode 3
065A    ED69       0975            OUT (C),H        ;I/O mask for Port B
                   0976                             ;
                   0977                             ;
                   0978                             ;
                   0979                             ;
                   0980                             ;
                   0981                             ;
                   0982   ;
065C    31A00F     0983   REF:     LD SP,CHPSTK     ;Initialize
                   0984                             ;stack pointer
065F    DD210008   0985            LD IX,REFIC      ;Initialize
                   0986                             ;reference IC
                   0987                             ;map pointer
0663    010000     0988            LD BC,0000H      ;Initialize
                   0989                             ;counter word
0666    CD8806     0990            CALL STORE       ;Generate the
                   0991                             ;reference table
0669    00         0992   ENDREF:  NOP
                   0993   ;
066A    31A00F     0994   UNKN:    LD SP,CHPSTK     ;Initialize
                   0995                             ;stack pointer
066D    DD21000C   0996            LD IX,UNKIC      ;Initialize un-
                   0997                             ;known IC map
                   0998                             ;pointer
0671    010000     0999            LD BC,0000H      ;Initialize
                   1000                             ;counter word
0674    CD8806     1001            CALL STORE       ;Generate the
                   1002                             ;unknown IC's
                   1003                             ;output table
                   1004   ;
0677    210008     1005   COMPAR:  LD HL,REFIC      ;Set-up for com-
                   1006                             ;pare using the
                   1007                             ;CPI instruction
067A    11000C     1008            LD DE,UNKIC      ;HL points to
                   1009                             ;ref table, DE
                   1010                             ;points to unk
                   1011                             ;IC table
067D    1A         1012   NEXTB:   LD A,(DE)        ;Load unknown
                   1013                             ;output byte into
                   1014                             ;accumulator
067E    EDA1       1015            CPI              ;Compare with (HL)
0680    2037       1016            JR NZ,BAD        ;If not =, we have
                   1017                             ;a bad IC
0682    13         1018            INC DE           ;If =, set up to.
                   1019                             ;test next bye
```

```
0683   EA7D06   1020           JP PE,NEXTB     ;If P/V flag = 1
                1021                           ;go test next byte
0686   1833     1022  GOOD:    JR START        ;If P/V flag = 0
                1023                           ;BC is zero and
                1024                           ;we have tested
                1025                           ;all the bytes
                1026  ;
0688   110000   1027  STORE:   LD DE,0000H     ;Initialize test
                1028                           ;word
068B   2A0300   1029  NTEST:   LD HL,(MASKW)   ;Load HL with mask
                1030                           ;word
068E   7B       1031           LD A,E          ;Perform 16-bit
                1032                           ;AND on mask and
                1033                           ;test words
068F   A5       1034           AND L
0690   6F       1035           LD L,A
0691   7A       1036           LD A,D
0692   A4       1037           AND H
0693   67       1038           LD H,A
0694   7C       1039  MASK:    LD A,H          ;Check if result
                1040                           ;of 16-bit AND = 0
0695   B5       1041           OR L
0696   201B     1042           JR NZ,NXTWD     ;If not 0, go to
                1043                           ;next test byte
                1044  ;
0698   7B       1045  TEST:    LD A,E          ;If = 0, it is a
                1046                           ;valid test word.
                1047                           ;Output it to IC
0699   D308     1048           OUT (08H),A
069B   7A       1049           LD A,D
069C   D309     1050           OUT (09H),A
069E   2A0300   1051           LD HL,(MASKW)   ;Get mask word for
                1052                           ;IC
06A1   DB08     1053           IN A,(08H)      ;Input LO byte
                1054                           ;from IC
06A3   A5       1055           AND L           ;Mask it
06A4   DD7700   1056           LD (IX),A       ;Store it
06A7   DD23     1057           INC IX          ;Update IX
06A9   DB09     1058           IN A,(09H)      ;Input HI byte
                1059                           ;from IC
06AB   A4       1060           AND H           ;Mask it
06AC   DD7700   1061           LD (IX),A       ;Store it
06AF   DD23     1062           INC IX          ;Update IX
06B1   03       1063           INC BC          ;Add two to
                1064                           ;counter
06B2   03       1065           INC BC
                1066  ;
06B3   13       1067  NXTWD:   INC DE          ;Get next test
                1068                           ;word
06B4   7A       1069           LD A,D
06B5   B3       1070           OR E
06B6   20D3     1071           JR NZ,NTEST     ;If DE is not
                1072                           ;zero, go back
                1073                           ;for next test
                1074                           ;word
```

```
06B8    C9          1075              RET                  ;If DE is zero
                    1076                                   ;full output
                    1077                                   ;table is gen-
                    1078                                   ;erated
06B9    1800        1079    BAD:      JR START             ;Bad IC, start
                    1080                                   ;over
                    1081    ;
06BB    18AD        1082    START:    JR UNKN              ;Jump to test
                    1083                                   ;routine for
                    1084                                   ;unknown IC
                    1085    ;
                    1086    ;
                    1087    ;
                    1088              NAME INITC1
06BD    ED5E        1089    INITC1:   IM2                  ;Z80 Interrupt Mode 2
06BF    21000F      1090              LD HL,TABLE          ;address of vector table
06C2    7C          1091              LD A,H               ;high byte of address
06C3    ED47        1092              LD I,A               ;set interrupt register
06C5    FD216E02    1093              LD IY,SERV1          ;service routine address
06C9    FD221A0F    1094              LD (TABLE+1AH),IY ;set in table
06CD    3E18        1095              LD A,18H             ;load interrupt vector
06CF    D310        1096              OUT (10H),A          ;to CTC CHANNEL 0
06D1    08          1097              EX AF,AF'            ;set format for CONVDI
06D2    3E40        1098              LD A,40H
06D4    08          1099              EX AF,AF'
06D5    3EC7        1100              LD A,0C7H            ;set channel control word
06D7    D311        1101              OUT (11H),A
06D9    3E05        1102              LD A,05H             ;set time constant
06DB    D311        1103              OUT (11H),A          ;register
06DD    C3C302      1104              JP MAIN              ;jump to routine MAIN
                    1105    ;
                    1106    ;
                    1107    ;
                    1108              NAME SERCT1
06E0    C5          1109    SERCT1:   PUSH BC              ;save status of BC
06E1    0E11        1110              LD C,11H             ;PORT 11H of CTC
06E3    C33104      1111              JP SERVI
                    1112    ;
                    1113    ;
                    1114    ;
                    1115              NAME SERCT2
06E6    C5          1116    SERCT2:   PUSH BC              ;save CPU registers
06E7    D5          1117              PUSH DE
06E8    E5          1118              PUSH HL
06E9    F5          1119              PUSH AF
06EA    DDE5        1120              PUSH IX
06EC    FDE5        1121              PUSH IY
06EE    FD2AE40F    1122              LD IY,(ADDL)         ;save state of (ADDL)
06F2    FDE5        1123              PUSH IY
06F4    0E16        1124              LD C,16H             ;input from CTC
06F6    ED40        1125              IN B,(C)
06F8    AF          1126              XOR A                ;clear A
06F9    90          1127              SUB B                ;find number of seconds
06FA    32E40F      1128              LD (ADDL),A          ;load ADDL with CTC data
06FD    DD23        1129    DST:      INC IX               ;update data stack pointer
```

```
06FF  DD23       1130             INC IX
0701  DD23       1131             INC IX
0703  00         1132             NOP                  ;no operation
0704  DD3600FF   1133             LD (IX+00H),0FFH     ;set DLOOPT time
0708  DD36010A   1134             LD (IX+01H),00AH     ;set CLOOPT time
070C  DD360202   1135   CLOOPT:   LD (IX+02H),02H      ;set DLOOPT time
0710  21E50F     1136             LD HL,ADDH           ;point to display buffer
0713  ED57       1137             LD A,I               ;find value of IFF2
0715  EA1C07     1138             JP PE,HIGHT
0718  3600       1139   LOWT:     LD (HL),00H          ;value = 0
071A  1802       1140             JR NEXTT
071C  3610       1141   HIGHT:    LD (HL),10H          ;value = 1
071E  ED73E20F   1142   NEXTT:    LD (DATAL),SP        ;copy SP to buffer
0722  21B90F     1143             LD HL,LEDL           ;set for CONVDI
0725  11E50F     1144             LD DE,ADDH           ;set for CONVDI
0728  CD7CFA     1145             CALL CONVDI
072B  CD09F9     1146   DLOOPT:   CALL DISPL
072E  DD3500     1147             DEC (IX+00)          ;timer for display
0731  20F8       1148             JR NZ,DLOOPT
0733  DD3502     1149             DEC (IX+02)          ;timer for display
0736  20F3       1150             JR NZ,DLOOPT
0738  DD3501     1151             DEC (IX+01)          ;timer for service routine
073B  20CF       1152             JR NZ,CLOOPT
073D  3E2F       1153             LD A,2FH             ;Channel 0 control word
073F  D314       1154             OUT (14H),A
0741  3E96       1155             LD A,96H             ;Channel 0 time constant
0743  D314       1156             OUT (14H),A
0745  3E47       1157             LD A,47H             ;Channel 1 control word
0747  D315       1158             OUT (15H),A
0749  3E40       1159             LD A,40H             ;Channel 1 time constant
074B  D315       1160             OUT (15H),A
074D  3E47       1161             LD A,47H             ;Channel 2 control word
074F  D316       1162             OUT (16H),A
0751  3E00       1163             LD A,00H             ;Channel 2 time constant
0753  D316       1164             OUT (16H),A
0755  3EC7       1165             LD A,0C7H            ;Channel 3 control word
0757  D317       1166             OUT (17H),A
0759  3E01       1167             LD A,01H             ;Channel 3 time constant
075B  D317       1168             OUT (17H),A
075D  FDE1       1169             POP IY               ;restore contents of ADDL
075F  FD22E40F   1170             LD (ADDL),IY
0763  FDE1       1171             POP IY               ;restore CPU registers
0765  DDE1       1172             POP IX
0767  F1         1173             POP AF
0768  E1         1174             POP HL
0769  D1         1175             POP DE
076A  C1         1176             POP BC
076B  FB         1177             EI                   ;enable interrupt flip-flop
076C  ED4D       1178             RETI                 ;return from interrupts
                 1179   ;
                 1180   ;
                 1181   ;
                 1182             NAME INITC3
076E  ED5E       1183   INITC3:   IM2                  ;Z80 Interrupt Mode 2
0770  21000F     1184             LD HL,TABLE          ;vector address table
```

```
0773  7C        1185              LD A,H           ;high byte of address
0774  ED47      1186              LD I,A           ;set interrupt register
0776  FD21E606  1187              LD IY,SERCT2     ;service routine address
077A  FD22260F  1188              LD (TABLE+26H),IY  ;set in table
077E  3E26      1189              LD A,26H         ;load interrupt vector
0780  D314      1190              OUT (14H),A      ;to CTC Channel 0
0782  08        1191              EX AF,AF'        ;set format for CONVDI
0783  3E40      1192              LD A,40H
0785  08        1193              EX AF,AF'
0786  3E2F      1194              LD A,2FH         ;Channel 0 control word
0788  D314      1195              OUT (14H),A
078A  3E96      1196              LD A,96H         ;Channel 0 time constant
078C  D314      1197              OUT (14H),A
078E  3E47      1198              LD A,47H         ;Channel 1 control word
0790  D315      1199              OUT (15H),A
0792  3E40      1200              LD A,40H         ;Channel 1 time constant
0794  D315      1201              OUT (15H),A
0796  3E47      1202              LD A,47H         ;Channel 2 control word
0798  D316      1203              OUT (16H),A
079A  3E00      1204              LD A,00H         ;Channel 2 time constant
079C  D316      1205              OUT (16H),A
079E  3EC7      1206              LD A,0C7H        ;Channel 3 control word
07A0  D317      1207              OUT (17H),A
07A2  3E01      1208              LD A,01H         ;Channel 3 time constant
07A4  D317      1209              OUT (17H),A
07A6  C3C302    1210              JP MAIN
                1211    ;
                1212    ;
                1213    ;
                1214              NAME INITC2
07A9  FD21E006  1215    INITC2:   LD IY,SERCT1     ;service routine address
07AD  FD22180F  1216              LD (TABLE+18H),IY  ;set in table
07B1  3EC7      1217              LD A,0C7H        ;channel 0 control word
07B3  D310      1218              OUT (10H),A
07B5  3E01      1219              LD A,01H         ;time constant register
07B7  D310      1220              OUT (10H),A      ;for channel 0
07B9  C3BD06    1221              JP INITC1
                1222    ;
                1223    ;
                1224    ;
07BC  (0010)    1225              DS 10H
                1226    ;
                1227    ;
                1228    ;
07CC            1229              ORG 0F000H
                1230              NAME BLKMVE
      (0100)    1231    ORIGIN    EQU 100H
      (0700)    1232    LENGHT    EQU 0700H
                1233    ;
                1234    ;
                1235    ;
F000  FB        1236    BLKMVE:   EI
F001  218EF0    1237              LD HL,RESTART
F004  F3        1238              DI
F005  110001    1239              LD DE,ORIGIN
```

```
F008  010007    1240                LD BC,LENGHT
F00B  EDB0      1241                LDIR
                1242  ;
                1243  ;
                1244  ;
                1245  ;
                1246  ;
                1247  ;
                1248                NAME NANOR2
F00D  DD21000C  1249  NANOR2:  LD IX,DSTACK     ;Set IX to RAM
                1250                                ;counter location
F011  2142F0    1251           LD HL,STRING
F014  11B80F    1252  MOVE:    LD DE,LEDH       ;and DE to dis-
                1253                                ;play buffer
F017  010A00    1254           LD BC,0AH        ;BC=no. of bytes
                1255                                ;to move
F01A  E5        1256           PUSH HL          ;Save character
                1257                                ;pointer
F01B  EDB0      1258           LDIR             ;Move first 10
                1259                                ;bytes
F01D  DD3600FF  1260           LD (IX),0FFH     ;Preset counter
F021  DD360101  1261           LD (IX+1H),01H   ;for display scan
                1262                                ;speed
F025  3E00      1263           LD A,00H
F027  32B80F    1264           LD (LEDH),A      ;Mask off LED dis-
                1265                                ;plays
F02A  32B90F    1266           LD (LEDH+1H),A
F02D  CD09F9    1267  DS:      CALL DISPL
F030  DD3500    1268           DEC (IX)         ;Time . . .
F033  20F8      1269           JR NZ,DS         ;   . . . delay
F035  DD3501    1270           DEC (IX+1H)      ;. . . and
F038  20F3      1271           JR NZ,DS         ;   . . . display
F03A  E1        1272           POP HL           ;Retrieve char-
                1273                                ;acter pointer
                1274                                ;value
F03B  23        1275           INC HL           ;And increment
F03C  7E        1276           LD A,(HL)        ;Check charac-
                1277                                ;ter for end-
                1278                                ;code
F03D  FE01      1279           CP 01H           ;'01', otherwise
                1280                                ;move along
F03F  20D3      1281           JR NZ,MOVE
F041  FF        1282           RST 38H          ;Return control
                1283                                ;to the Nanocomputer
                1284                                ;operating system
                1285  ;
                1286  ;
                1287  ;
F042  00000000  1288  STRING:  DB 00H,00H,00H,00H      ;Leading blanks
F046  00000000  1289           DB 00H,00H,00H,00H
F04A  0000B6BC  1290           DB 00H,00H,0B6H,0BCH    ;SG
F04E  B602EE1E  1291           DB 0B6H,02H,0EEH,1EH    ;S-AT
F052  9EB600EC  1292           DB 9EH,0B6H,00H,0ECH    ;ES N
F056  EEECFC00  1293           BD 0EEH,0ECH,0FCH,00H   ;AN0
F05A  0A3A381E  1294           DB 0AH,3AH,38H,1EH      ;ROUT
```

```
F05E   202A9EB6   1295            DB 20H,2AH,9EH,0B6H   ;INES
F062   000A9E1C   1296            DB 00H,0AH,9EH,1CH    ;  REL
F066   9EEEB69E   1297            DB 9EH,0EEH,0B6H,9EH  ;EASE
F06A   00DA02DA   1298            D8 00H,0DAH,02H,0DAH  ;  2-2
F06E   001CFCEE   1299            DB 00H,1CH,0FCH,0EEH  ;  LOA
F072   7A9E7A00   1300            DB 7AH,9EH,7AH,00H    ;DED
F076   00000000   1301            DB 00H,00H,00H,00H    ;
F07A   009C60EE   1302            DB 00H,9CH,60H,0EEH   ;  CIA
F07B   FC000010   1303            DB 0FCH,00H,00H,10H   ;O  -
F082   00100110   1304            DB 00H,10H,01H,10H    ;  -  -
F086   00000000   1305            DB 00H,00H,00H,00H    ;Trailing blanks
F08A   00000000   1306            DB 00H,00H,00H,00H
                  1307  RESTART:
                  1308  ;
                  1309  ;
                  1310  ;
```

Anhang C

Regeln für den Umgang mit MOS-ICs

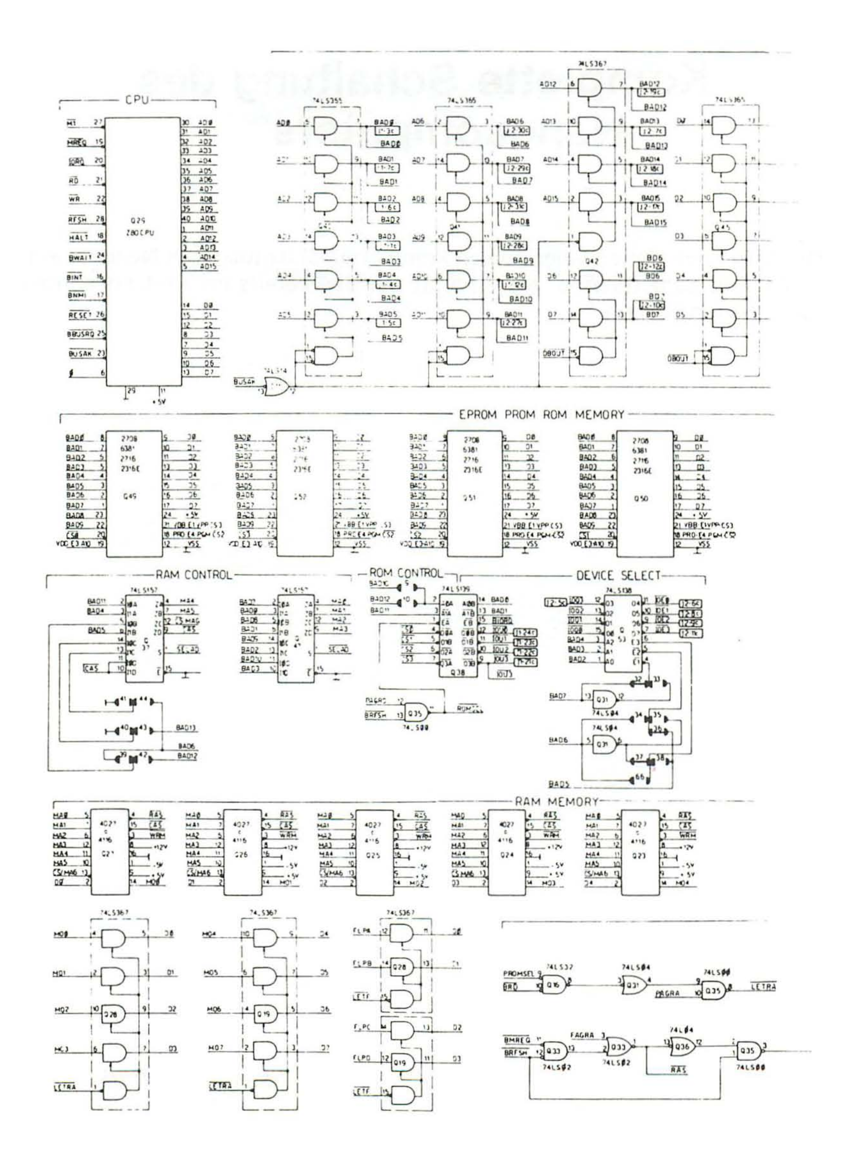
MOS-ICs sind sehr empfindlich und können leicht zerstört werden:
- durch statische Aufladung;
- durch falsches Einsetzen in die IC-Fassungen auf der Nanocomputer-Platine.

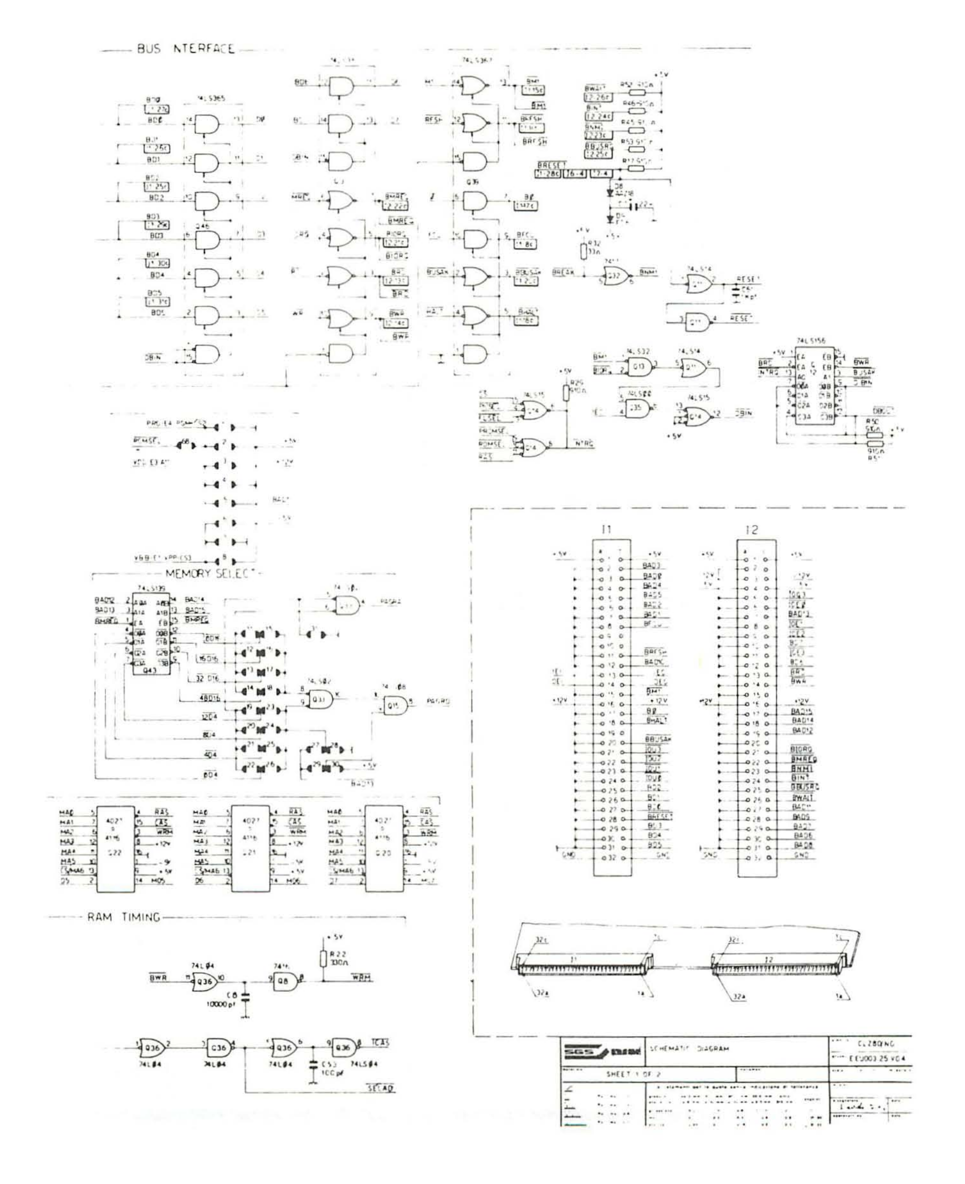
Deshalb sind beim Umgang mit MOS-ICs folgende Vorsichtsmaßnahmen notwendig:
1. Sorgen Sie dafür, daß Ihr Körper nicht statisch geladen ist. Die statische Körperladung kann man durch Abreiben der Hände mit einem leitenden Material reduzieren.
2. Vermeiden Sie jeden direkten körperlichen Kontakt mit den Anschlußpins.
3. Vermeiden Sie jede Berührung der Anschlußpins mit statisch aufgeladenem Material, z.B. Nylon. Dazu gehört auch die Lagerung der ICs in Kunststoff-Materialkästen.
4. Für den Transport sollten die Anschlußpins in einem leitenden Schaumstoff stecken; es sei denn, daß sie in der IC-Fassung auf der Platine stecken. Nur diese beiden Transportbedingungen halten die statischen Ladungen von den Anschlußpins fern.
5. Beim Einstecken der ICs in die Fassung, muß man auf die richtige Lage der ICs achten: Pin 1 an Anschlußpunkt 1, Pin 2 an Anschlußpunkt 2 . . . . . usw.

Anhang **D**

Komplette Schaltung des Nanocomputers®

Die folgenden Seiten zeigen einen kompletten Schaltplan des Nanocomputers®. Das Schaltungskonzept stimmt mit den bereits im Text erwähnten Einzelheiten überein.


CPU
Q29
Z80 CPU
EPROM PROM ROM MEMORY
RAM CONTROL
ROM CONTROL
DEVICE SELECT
RAM MEMORY


BUS INTERFACE
MEMORY SELECT
RAM TIMING
SCHEMATIC DIAGRAM
SHEET 1 OF 2

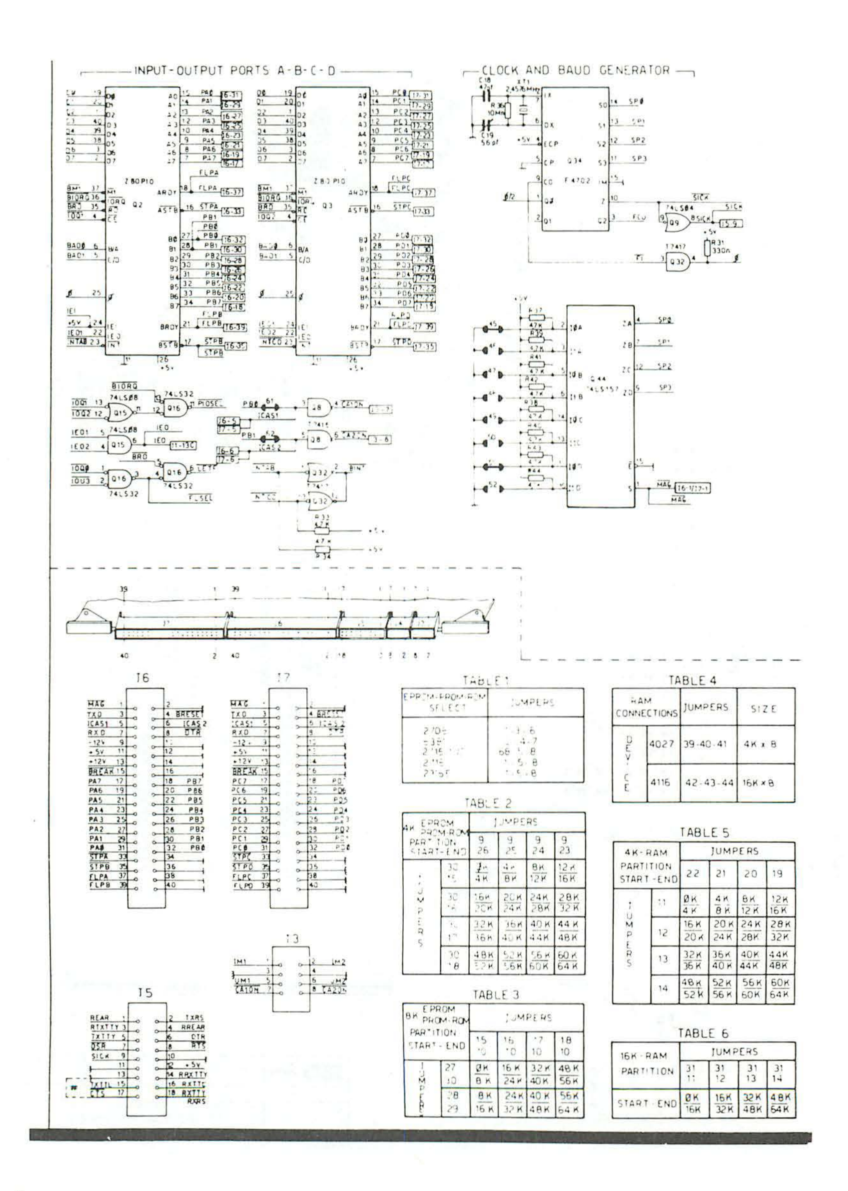


TABLE 1

EPROM-PROM-ROM SELECT	JUMPERS
2708	[illegible] 6
[illegible]	[illegible] 4-7
[illegible]	68 [illegible] 8
[illegible]	[illegible] 5 8
[illegible]	[illegible] 8

TABLE 2

4K EPROM PROM-ROM PARTITION START-END		JUMPERS 9/26	9/25	9/24	9/23
JUMPERS	30/[illegible]	[illegible] / 4K	4K / 8K	8K / 12K	12K / 16K
	30/[illegible]	16K / 20K	20K / 24K	24K / 28K	28K / 32K
	[illegible]	32K / 36K	36K / 40K	40K / 44K	44K / 48K
	30/18	48K / 52K	52K / 56K	56K / 60K	60K / 64K

TABLE 3

8K EPROM PROM-ROM PARTITION START-END		JUMPERS 15/10	16/10	17/10	18/10
JUMPERS	27/10	ØK / 8K	16K / 24K	32K / 40K	48K / 56K
	28/29	8K / 16K	24K / 32K	40K / 48K	56K / 64K

TABLE 4

RAM CONNECTIONS		JUMPERS	SIZE
DEVICE	4027	39-40-41	4K x 8
	4116	42-43-44	16K x 8

TABLE 5

4K-RAM PARTITION START-END		JUMPERS 22	21	20	19
JUMPERS	11	ØK / 4K	4K / 8K	8K / 12K	12K / 16K
	12	16K / 20K	20K / 24K	24K / 28K	28K / 32K
	13	32K / 36K	36K / 40K	40K / 44K	44K / 48K
	14	48K / 52K	52K / 56K	56K / 60K	60K / 64K

TABLE 6

16K-RAM PARTITION	JUMPERS 31/11	31/12	31/13	31/14
START-END	ØK / 16K	16K / 32K	32K / 48K	48K / 64K

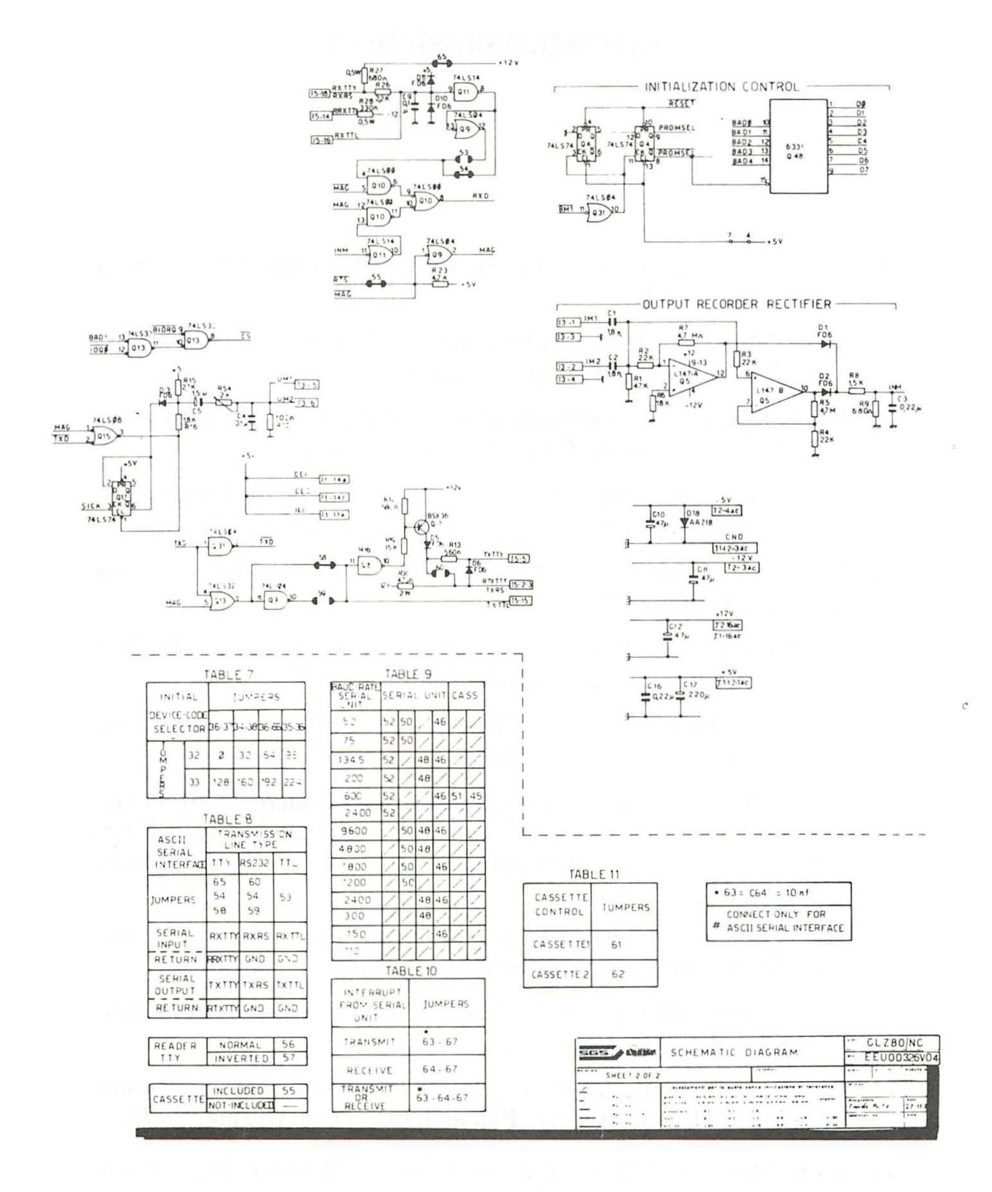


TABLE 7

INITIAL DEVICE-CODE SELECTOR		JUMPERS 36-37	34-38	36-65	35-36
JUMPERS	32	Ø	32	64	96
	33	128	160	192	224

TABLE 8

ASCII SERIAL INTERFACE	TRANSMISSION LINE TYPE TTY	RS232	TTL
JUMPERS	65 54 58	60 54 59	53
SERIAL INPUT	RXTTY	RXRS	RXTTL
RETURN	RRXTTY	GND	GND
SERIAL OUTPUT	TXTTY	TXRS	TXTTL
RETURN	RTXTTY	GND	GND

READER TTY	NORMAL	56
	INVERTED	57

CASSETTE	INCLUDED	55
	NOT-INCLUDED	—

TABLE 9

BAUD RATE SERIAL UNIT	SERIAL UNIT				CASS	
50	52	50	/	46	/	/
75	52	50	/	/	/	/
134.5	52	/	48	46	/	/
200	52	/	48	/	/	/
600	52	/	/	46	51	45
2400	52	/	/	/	/	/
9600	/	50	48	46	/	/
4800	/	50	48	/	/	/
1800	/	50	/	46	/	/
1200	/	50	/	/	/	/
2400	/	/	48	46	/	/
300	/	/	48	/	/	/
150	/	/	/	46	/	/
110	/	/	/	/	/	/

TABLE 10

INTERRUPT FROM SERIAL UNIT	JUMPERS
TRANSMIT	• 63 - 67
RECEIVE	64 - 67
TRANSMIT OR RECEIVE	• 63 - 64 - 67

TABLE 11

CASSETTE CONTROL	JUMPERS
CASSETTE 1	61
CASSETTE 2	62

Anhang **E**

Literaturverzeichnis

1. *The Compact Edition of the Oxford English Dictionary,* Oxford Univ. Press, 1971.

2. Rudolf F. Graf, *Modern Dictionary of Electronics,* Howard W. Sams & Co., Inc., Indianapolis, Indiana, 1977.

3. James Martin, *Telecommunications and the Computer,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.

4. Abraham Marcus and John D. Lenk, *Computers for Technicians,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

5. Microdata Corporation, *Microprogramming Handbook,* Santa Ana, California, 1971.

6. J. Blukis and M. Baker, *Practical Digital Electronics,* Hewlett-Packard Company, Santa Clara, California, 1974.

7. Donald E. Lancaster, *TTL Cookbook,* Howard W. Sams & Co., Inc., Indianapolis, Indiana, 1974.

8. H. V. Malmstadt, C. G. Enke, and S. R. Crouch, *Instrumentation for Scientists Series, Module 3. Digital and Analog Data Conversions,* W. A. Benjamin, Inc., Menlo Park, California, 1973-4.

9. H. V. Malmstadt and C. G. Enke, *Digital Electronics for Scientists,* W. A. Benjamin, Inc., New York, 1969.

10. J. D. Lenk, *Handbook of Logic Circuits,* Reston Publishing Company, Inc., Reston, Virginia, 1972.

11. A. James Diefenderfer, *Principles of Electronic Instrumentation,* W. B. Saunders Company, Philadelphia, Pennsylvania, 1972.

12. P. R. Rony and D. G. Larsen, *Logic & Memory Experiments*

Using TTL Integrated Circuits, Book 2, Howard W. Sams & Co., Inc., Indianapolis, Indiana, 1979.

13. Robert L. Morris and John R. Miller, Editors, *Designing with TTL Integrated Circuits,* McGraw-Hill Book Company, New York, 1971.
14. Charles J. Sippl, *Microcomputer Dictionary and Guide,* Matrix Publishers, Inc., Champaign, Illinois, 1976.
15. Donald Eadie, *Introduction to the Basic Computer,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.
16. Texas Instruments Incorporated, *Microprocessor Handbook,* Dallas, Texas, 1975.

Sachwortverzeichnis

Q

R

S

T

U

V

W

Z

Mehr über das Nanocomputer-Trainingssystem

Das Nanocomputer-Trainingssystem von SGS-ATES macht Sie mit der Programmierung des Z-80 Mikrocomputers vertraut. Die modulare Bauweise gestattet den schrittweisen Auf- und Ausbau einer kompletten Mikrocomputer-Familie. Die drei Grundbausteine sind:

Hardware

NBZ80 — Ein leistungsstarker "Ein-Platinen-Computer". Zum Nanocomputer ® gehört eine Ein- und Ausgabe-Einheit (hexadezimal und 7-Segment-Display) sowie eine 2k-Byte-Speichereinheit. Mit dem zusätzlichen Kit KNZ80 wird aus dem Nanocomputer ® ein Mikrocomputer CLZ80, der die höheren Programmiersprachen Assembler und BASIC versteht.

NEZ80 — Die Experimentierplatine NEZ80 ist für den Anschluß mit dem NBZ80-Signalbus vorbereitet. Die Platine ermöglicht den lötfreien Aufbau einer beliebigen Digitalschaltung.

NPZ80 — Das separate Netzteil liefert die erforderlichen Versorgungsspannungen ± 5 V und ± 12 V.

Experimentier-Kits

Für die Durchführung sämtlicher, in allen drei Büchern beschriebenen Experimente sind die Experimentier-Kits K1Z80 und K2Z80 erforderlich.

Bücher

Buch 1 — beschreibt die Mikrocomputer-Grundausführung, die Maschinensprache, die Assemblersprache sowie über 30 Demonstrationsprogramme.

Buch 2 — ist eine Einführung in die Digitaltechnik mit der T74LSxx-TTL-Familie (Low Power Schottky TTL). Funktionsbeschreibungen und praktische Experimente machen den Umgang mit diesen ICs deutlich.

Buch 3 — behandelt Interface-Schaltungen für den Z80: Speichereinheiten, Parallel-Ein- und -Ausgabe (PIO), CTC-Schaltungen (Counter-Timer-Circuit). Viele Experimente (dafür sind ca. 2k-Byte-Speicherplatz erforderlich) ergänzen das Buch zu einem praxisbezogenen Nachschlagewerk.

Die Tabelle zeigt, zu welchem Buch welcher Experimentiersatz notwendig ist.

	NBZ80	NEZ80	NPZ80	K1Z80	K2Z80
Buch 1	X		X		
Buch 2		X	X	X	X
Buch 3	X	X	X	X	X

Das modulare Aufbausystem strapaziert Ihren Hobby-Etat nicht zu sehr. Sie können schrittweise -von Buch 1 bis zu Buch 3- das Gerät aufbauen. Mehr Informationen über Lieferbarkeit, Preis usw. geben die SGS-Ates-Verkaufbüros. Eine entsprechende Liste finden Sie auf der folgenden Seite.

EG

S1

u_1

u_2

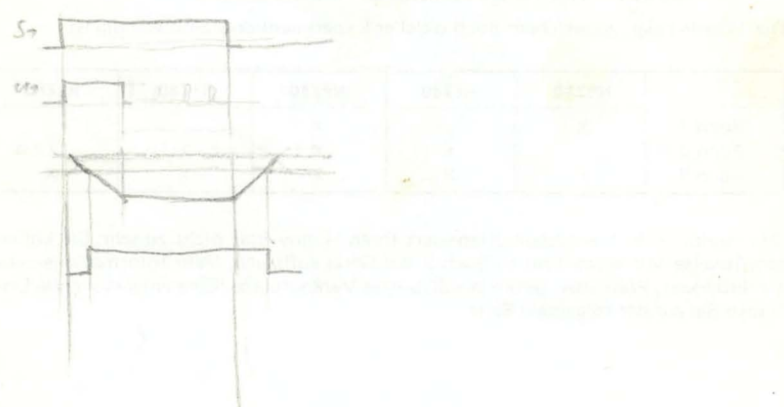
S2

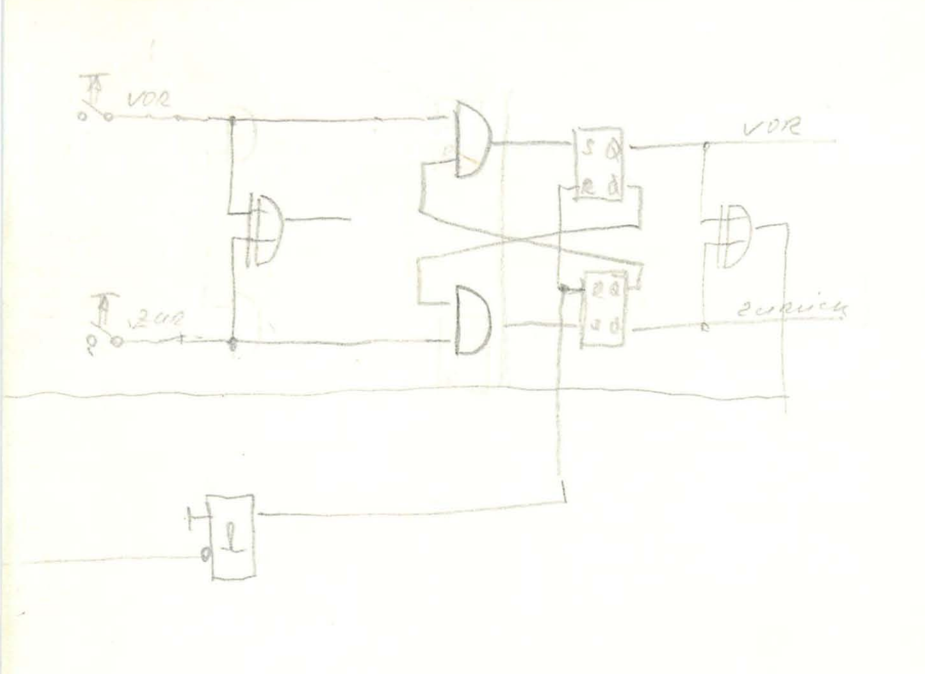
Freigabe für Tasten im Hand

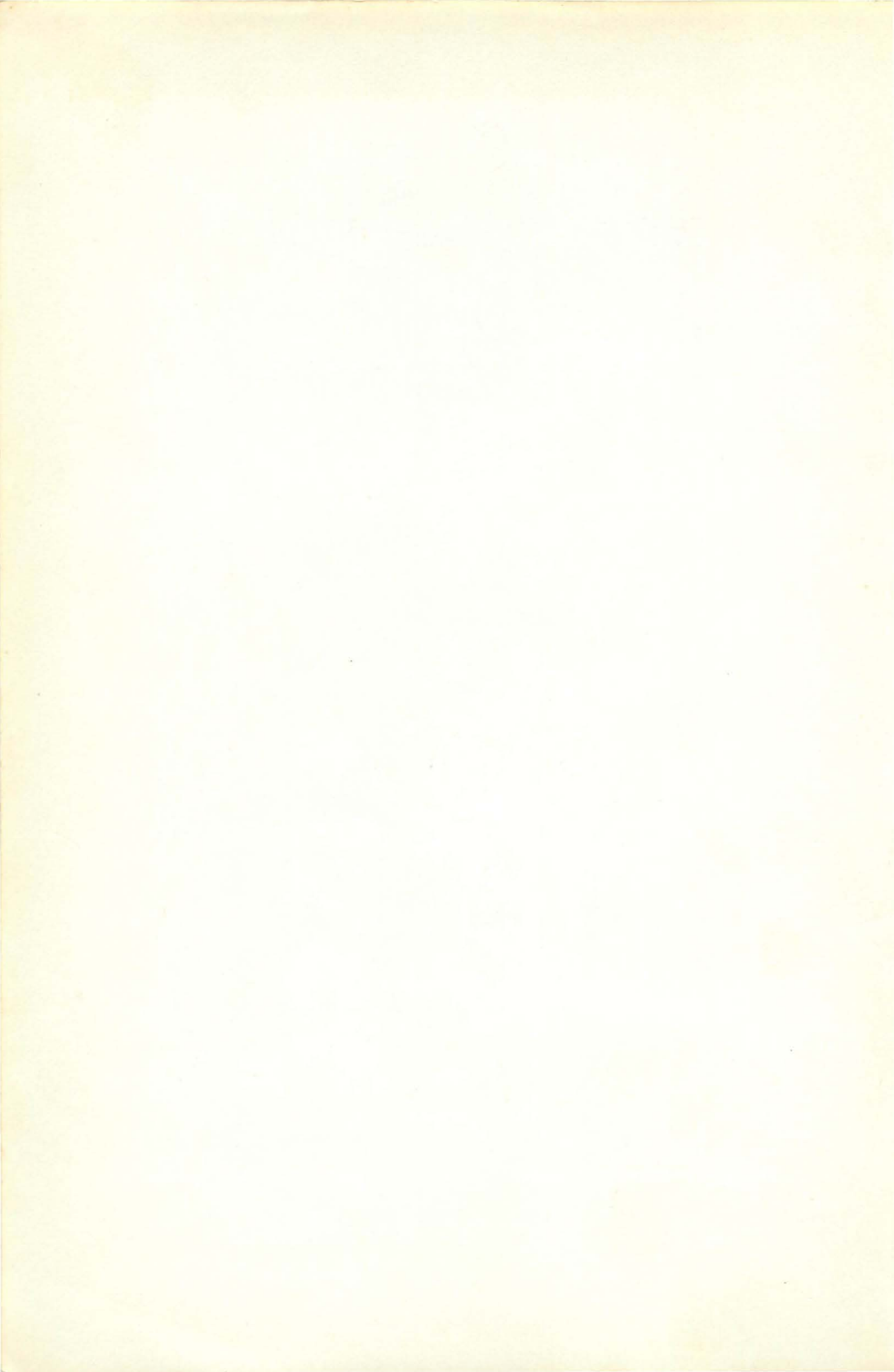
EG —

SG —

VON FREIG
ALS 770




VOR
zur
VOR
zurück



NBZ80-Nanocomputer®-Vertriebsorganisation

INTERNATIONAL HEADQUARTER
SGS-ATES
Componenti Elettronici SpA
Via C. Olivetti 2
I - 20041 Agrate Brianza
Italy
Tel.: 0039 - 39 - 650341-4
650441-5
Telex: 0043 - 330131 - 141

Verkauf und Verwaltung:
Haidling 17
Postfach 1180
8018 Grafing bei München
Telefon: (0 80 92) 691
Telex: 05 27 378

Verkaufsbüros:

SGS-ATES Deutschland GmbH
Verkaufsbüro
Gatower Straße 185
1000 Berlin 20
Telefon: (030) 3 62 20 31-32
Telex: 01 85 418

SGS-ATES Deutschland GmbH
Verkaufsbüro
Hubertusstraße 7
3012 Langenhagen
Telefon: (05 11) 77 20 75-77
Telex: 09 23 195

SGS-ATES Deutschland GmbH
Verkaufsbüro
Kalifenweg 45
7000 Stuttgart 80
Telefon: (07 11) 71 30 91-92
Telex: 07 255 545

SGS-ATES Deutschland GmbH
Verkaufsbüro
Landsberger Straße 289
8000 München 21
Telefon: (089) 58 20 47-48
Telex: 05 215 784

SGS-ATES Deutschland GmbH
Verkaufsbüro
Parsifalstraße 10
8500 Nürnberg 15
Telefon: (09 11) 4 96 45-46
Telex: 06 26 243

Vertragshändler:

Getronic-Gebensleben
Warnstedtstraße 59
2000 Hamburg 54
Telefon: (040) 540 40 46
Telex: 02 15 032

Ingenieurbüro
Karl-Heinz Dreyer
Flensburger Straße 3
2380 Schleswig
Telefon: (0 46 21) 2 31 21
Telex: 02 21 334

Setron
Schiffer-Elektronik
GmbH & Co. KG
Leopoldstraße 29
3300 Braunschweig
Postfach 46 29
Telefon: (05 31) 4 65 32
Telex: 09 52 812

Gesco Electronic GmbH
Postfach 4205
4902 Bad Salzuflen 1
Telefon: (05 222) 83 353
Telex: 09 31 21 98

Weisbauer Elektronik GmbH
Heiliger Weg 1
4600 Dortmund
Telefon (02 31) 57 95 47
Telex: 82 25 38

Siegfried Ecker
Königsberger Straße 2
6120 Michelstadt
Postfach 33 44
Telefon: (0 60 61) 22 33
Telex: 04 191 630

elecdis ruggaber GmbH
Hertichstraße 41
7250 Leonberg/Eltingen
Telefon: (0 71 52) 4 70 81
Telex: 07 24 192

Electronic 2000
Neumarkter Straße 75
8000 München 80
Telefon: (089) 43 40 61
Telex: 52 25 61

MBS-Electronic GmbH
Benzstr. 1
Postfach 1111
8011 Kirchheim
Telefon: (089) 90 37 181
Telex: 05 215 555

Gustav Back KG
Handelsabteilung für
elektronische Bauelemente
Elterdorfer Straße 7
8500 Nürnberg 15
Postfach 15 02 80
Telefon: (09 11) 3 49 66
Telex: 06 22 334

ÖSTERREICH
Burisch GmbH & Co. KG
Postfach 24
A - 1215 Wien
Telefon: (02 22) 38 76 38
Telex: 01 - 13 310

Z-80

Interface-Technik und Anwendung

Interface – Grenzfläche (zweier Körper), Zwischenschicht, Trennfläche, Kopplungselektronik, Anschluß . . . – das sind einige Definitionen, die ein Wörterbuch unter dem Suchwort "interface" gibt. Das Stichwort ist Kopplungselektronik. Sie ist das Bindeglied zwischen dem Anwender und der CPU. Das Wissen um die Programmierung allein reicht nicht aus, um den Mikroprozessor optimal zu nutzen. Das ist erst zusammen mit dem richtigen Verständnis um die Interface-Technik und deren Anwendung möglich.
Das vorliegende Buch befaßt sich ebenso intensiv mit der Interface-Technik und deren Anwendung, wie Buch 1 die Programmierung beschreibt. In diesem Buch sammeln Sie Hintergrundinformation für den Umgang mit dem Z-80. Neben anderen Digital-Bausteinen ist der PIO (Parallele Ein-/Ausgabeeinheit) Mittelpunkt des Interesses. Nach der theoretischen Einführung je Interface -Detail folgt die Vertiefung des Erlernten im praktischen Experiment. Mit kleinen Programmen und Experimentier-Schaltungen wiederholt der Nanocomputer® NBZ80S (von SGS-ATES) die besprochenen Details. Schritt für Schritt zeigt der NBZ80S die praktische Wirkung. Diese Arbeitsmethode macht Sie mit dem Z-80 so vertraut, daß Sie das erworbene Wissen für Hobby und Beruf professionell nutzen können.

Elektor Verlag GmbH, D-5133 Gangelt 1

ISBN 3-921608-17-1

DAZ80NINTFG/1